ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΠΤΥΞΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΓΙΑ ΤΗ ΣΧΕΔΙΑΣΗ ΠΕΠΕΡΑΣΜΕΝΩΝ ΑΥΤΟΜΑΤΩΝ

ΓΕΩΡΓΟΣΟΥΛΗΣ ΕΥΑΓΓΕΛΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΤΖΙΑΛΛΑΣ ΓΡΗΓΟΡΙΟΣ
ΚΑΘΗΓΗΤΗΣ

Λαμία, 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΠΤΥΞΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΓΙΑ ΤΗ ΣΧΕΔΙΑΣΗ ΠΕΠΕΡΑΣΜΕΝΩΝ ΑΥΤΟΜΑΤΩΝ

ΓΕΩΡΓΟΣΟΥΛΗΣ ΕΥΑΓΓΕΛΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΤΖΙΑΛΛΑΣ ΓΡΗΓΟΡΙΟΣ
ΚΑΘΗΓΗΤΗΣ

Λαμία, 2025

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

DEVELOPMENT OF A WEB APPLICATION FOR THE DESIGN OF FINITE AUTOMATA

GEORGOSOULIS EVANGELOS

FINAL THESIS

ADVISOR

TZIALLAS GRIGORIOS
PROFESSOR

Lamia, 2025

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: 30/06/2025

Ο Δηλών

Γεωργοσούλης Ευάγγελος

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία επικεντρώνεται στη δημιουργία μιας διαδραστικής διαδικτυακής εφαρμογής για την σχεδίαση και προσομοίωση πεπερασμένων αυτομάτων, τόσο ντετερμινιστικών (DFA) όσο και μη ντετερμινιστικών (NFA) μοντέλων. Η εφαρμογή επιτρέπει στον χρήστη να δημιουργεί καταστάσεις και μεταβάσεις, να εισάγει συμβολοσειρές προς έλεγχο αποδοχής και να αποθηκεύει ή να φορτώνει αυτόματα, είτε τοπικά είτε μέσω cloud. Αποτελεί μία στατική ιστοσελίδα (Single Page Application - SPA) με οδηγίες χρήσης μέσω αναδυόμενων παραθύρων (modals). Έχει αναπτυχθεί με HTML, CSS, JavaScript, Node.js και PostgreSQL.

Η εργασία παρουσιάζει το θεωρητικό υπόβαθρο των πεπερασμένων αυτομάτων, συγκρίνει υπάρχουσες λύσεις σχεδίασης και τεκμηριώνει αναλυτικά τον σχεδιασμό και την υλοποίηση του συστήματος. Η εφαρμογή απευθύνεται κυρίως σε φοιτητές που μελετούν Θεωρία Υπολογισμού, καθώς και σε εκπαιδευτικούς που αναζητούν ένα εύχρηστο και εκπαιδευτικά αποτελεσματικό εργαλείο παρουσίασης και κατανόησης αυτομάτων.

ABSTRACT

This thesis focuses on the development of an interactive web application for the design and simulation of finite automata, including both deterministic (DFA) and nondeterministic (NFA) models. The application enables users to construct states and transitions, input strings for acceptance testing and save or load automata either locally or through a server-based cloud storage system. It is implemented as a Single Page Application (SPA), offering user guidance via modal dialogs and is built using modern web technologies such as HTML, CSS, JavaScript, Node.js and PostgreSQL.

The thesis presents the theoretical background of finite automata, reviews and compares existing design tools and thoroughly documents the architecture and implementation of the system. The application is primarily intended for students studying automata theory, as well as for educators seeking a practical and effective tool for demonstrating and understanding finite automata.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον επιβλέποντα καθηγητή, Τζιάλλα Γρηγόριο, για την καθοδήγηση, τη διαθεσιμότητα και την άριστη συνεργασία καθ' όλη τη διάρκεια της εκπόνησης της πτυχιακής εργασίας.

Ευχαριστώ επίσης την οικογένειά μου για την συνεχή στήριξη, την πίστη στις δυνατότητές μου και την κατανόηση που επέδειξε σε όλα τα στάδια των σπουδών μου.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου για τις ιδέες, την ενθάρρυνση και τη στήριξή τους στις στιγμές που ήταν περισσότερο απαραίτητη.

Table of Contents

1. ΕΙΣΑΓΩΓΗ

1.1 Σκοπός και Περιγραφή Εργασίας

Η παρούσα πτυχιακή εργασία έχει ως σκοπό την ανάπτυξη μιας διαδραστικής εφαρμογής για τη σχεδίαση, ανάλυση και προσομοίωση πεπερασμένων αυτομάτων (Finite Automata), τόσο ντετερμινιστικών (DFA) όσο και μη-ντετερμινιστικών (NFA). Μέσα από ένα φιλικό προς τον χρήστη περιβάλλον, η εφαρμογή επιτρέπει τη δημιουργία αυτομάτων καθώς και τον έλεγχο αποδοχής συμβολοσειρών, δίνοντας τη δυνατότητα αποθήκευσης και φόρτωσης τοπικά ή μέσω server.

Η σημασία της εργασίας έγκειται στην ενίσχυση της εκπαιδευτικής διαδικασίας στη Θεωρία Υπολογισμού, προσφέροντας ένα εύχρηστο εργαλείο για φοιτητές και εκπαιδευτικούς που επιθυμούν να πειραματιστούν με αυτόματα και να κατανοήσουν καλύτερα τις έννοιες μέσω οπτικής αναπαράστασης και άμεσης αλληλεπίδρασης. Η εφαρμογή καλύπτει ένα σημαντικό κενό μεταξύ θεωρίας και πράξης, επιτρέποντας στον χρήστη όχι μόνο να κατασκευάζει αυτόματα, αλλά και να δοκιμάζει διάφορες συμβολοσειρές σε πραγματικό χρόνο, παρατηρώντας έτσι τα αποτελέσματα για να διαπιστωθεί εάν το σχεδιασμένο αυτόματο είναι σωστά δομημένο.

Επιπλέον, η υλοποίηση της εφαρμογής με τεχνολογίες διαδικτύου (HTML, CSS, JavaScript, Node.js, PostgreSQL) ενισχύει τη δυνατότητα πρόσβασης από διαφορετικές πλατφόρμες, χωρίς την ανάγκη εγκατάστασης λογισμικού, καθιστώντας την διαθέσιμη σε ευρύτερο κοινό. Μέσα από τη χρήση σύγχρονων τεχνολογιών, επιδιώκεται η δημιουργία μιας σύγχρονης, επεκτάσιμης και χρηστικής λύσης, που μπορεί να αποτελέσει τη βάση για περαιτέρω επεκτάσεις, όπως η υποστήριξη άλλων τύπων αυτομάτων ή γραμματικών.

Τέλος, η εργασία αυτή συμβάλλει στην ανάπτυξη δεξιοτήτων προγραμματισμού, σχεδιασμού συστημάτων και υλοποίησης εκπαιδευτικών εργαλείων, συνδέοντας θεωρητικές γνώσεις με πρακτικές εφαρμογές σε έναν τομέα με ιδιαίτερη σημασία για την επιστήμη των Υπολογιστών.

1.2 Ορισμός Περιοχής Μελέτης

Η παρούσα πτυχιακή εργασία εντάσσεται στη θεματική περιοχή της Θεωρίας Υπολογισμού, ενός βασικού κλάδου της Επιστήμης Υπολογιστών που μελετά τα υπολογιστικά μοντέλα και τα θεωρητικά όρια του τι μπορεί να υπολογιστεί με τη χρήση τέτοιων μηχανών. Ειδικότερα, η εργασία επικεντρώνεται στη μελέτη και ανάλυση των πεπερασμένων αυτομάτων, τα οποία αποτελούν θεμελιώδη μοντέλα υπολογισμού για τον ορισμό και την αναγνώριση κανονικών γλωσσών.

Τα Πεπερασμένα Αυτόματα (ΠΑ) χρησιμοποιούνται ευρέως στην Επιστήμη των Υπολογιστών για την επίλυση προβλημάτων, όπως η λεκτική ανάλυση (lexical analysis), η ανάλυση κανονικών εκφράσεων (Regular Expressions) και η δημιουργία μεταγλωττιστών (compilers). Η μελέτη τους συμβάλλει στην κατανόηση θεμελιωδών εννοιών υπολογισμού, όπως η ντετερμινιστικότητα, η μη ντετερμινιστικότητα, η αποδοχή γλωσσών και η περιγραφική ισχύς διαφορετικών υπολογιστικών μοντέλων.

Στο πλαίσιο αυτής της εργασίας, η περιοχή μελέτης επεκτείνεται επίσης στον σχεδιασμό και την ανάπτυξη λογισμικού που επιτρέπει τη διαδραστική αναπαράσταση και προσομοίωση πεπερασμένων αυτομάτων μέσω διαδικτυακής εφαρμογής. Ο συνδυασμός θεωρητικών εννοιών και τεχνολογιών διαδικτύου αποτελεί βασικό άξονα της εργασίας, με στόχο τη δημιουργία ενός εργαλείου που γεφυρώνει τη θεωρία με την πράξη, ενισχύοντας παράλληλα τη μαθησιακή διαδικασία μέσω της πρακτικής ενασχόλησης.

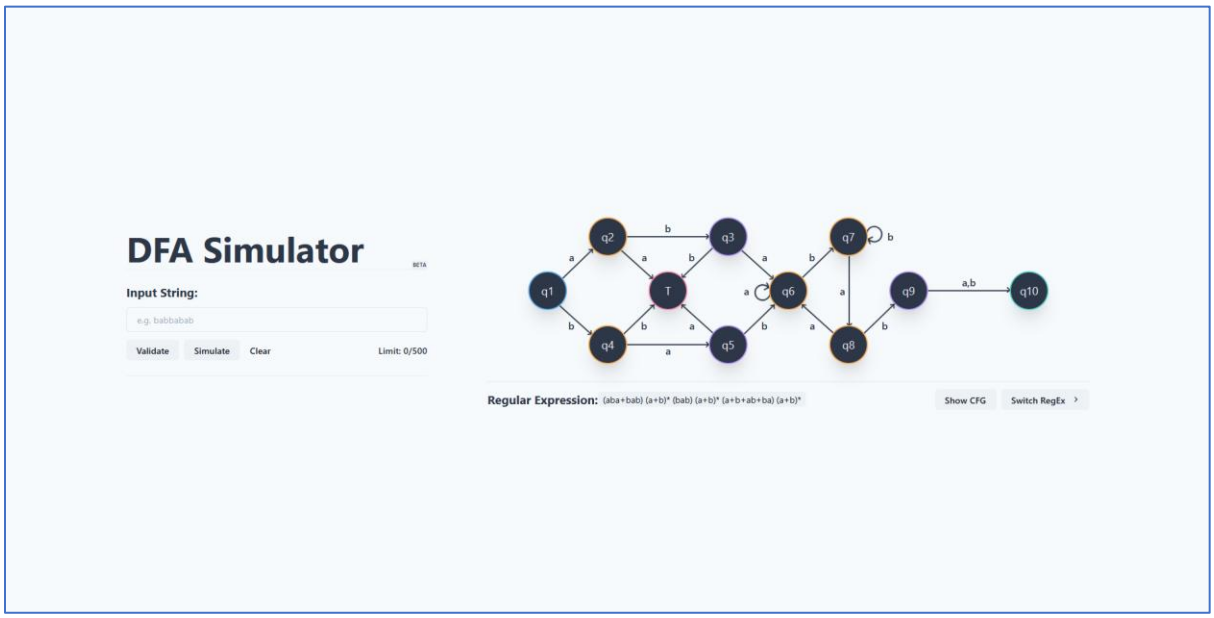
Μέσα από αυτή την περιοχή μελέτης, η εργασία στοχεύει στη διερεύνηση μεθόδων που καθιστούν πιο προσβάσιμη και κατανοητή την έννοια των αυτομάτων για φοιτητές, εκπαιδευτικούς και ενδιαφερόμενους για τη Θεωρία Υπολογισμού, μέσα από ένα περιβάλλον εύχρηστο, διαδραστικό και λειτουργικό.

2. ΕΦΑΡΜΟΓΕΣ ΓΙΑ ΣΧΕΔΙΑΣΗ ΑΥΤΟΜΑΤΩΝ

2.1 Παρόμοια Εργαλεία

Στο πλαίσιο της πτυχιακής εργασίας πραγματοποιήθηκε έρευνα σχετικά με υπάρχουσες διαδικτυακές εφαρμογές και εργαλεία που παρέχουν δυνατότητες σχεδίασης και προσομοίωσης πεπερασμένων αυτομάτων. Ορισμένα από τα πιο γνωστά εργαλεία που εντοπίστηκαν είναι τα εξής:

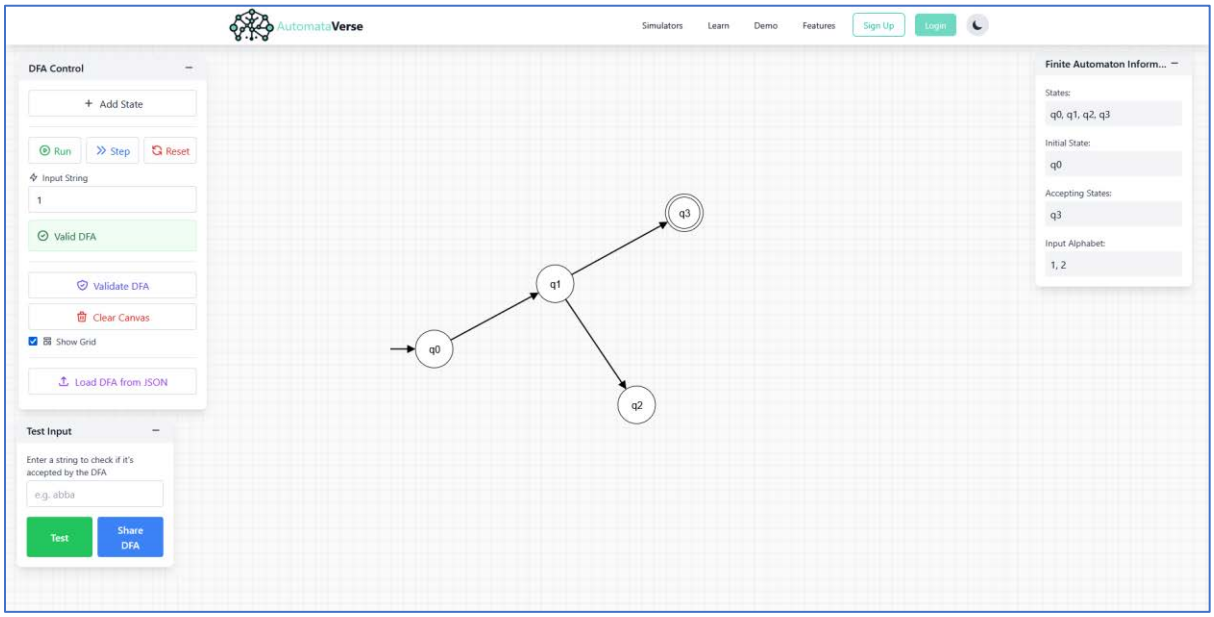
- DFA Simulator (https://dfa-simulator.vercel.app)



Εικόνα 1. Διεπαφή του εργαλείου DFA Simulator

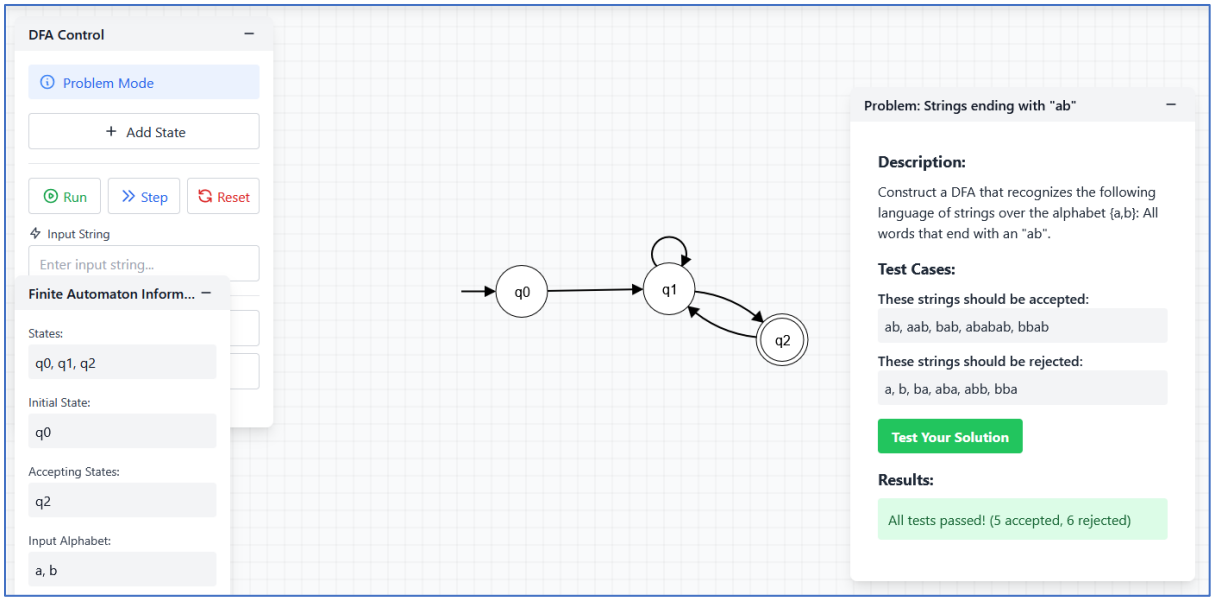
Πρόκειται για μια απλή και γρήγορη διαδικτυακή εφαρμογή που επιτρέπει στον χρήστη να πειραματιστεί με έτοιμα ντετερμινιστικά πεπερασμένα αυτόματα (ΝΠΑ), τα οποία δημιουργούνται αυτόματα μέσω της επιλογής “Switch RegEx”. Ο χρήστης δεν έχει τη δυνατότητα να σχεδιάσει από την αρχή δικά του αυτόματα ή να επεξεργαστεί τις καταστάσεις και τις μεταβάσεις τους. Ωστόσο, μπορεί να εισάγει συμβολοσειρές εισόδου για να ελέγξει αν γίνονται αποδεκτές ή να παρακολουθήσει βήμα-βήμα την πορεία της προσομοίωσης. Παρέχεται επίσης η δυνατότητα εμφάνισης της κανονικής έκφρασης και της αντίστοιχης γραμματικής που αντιστοιχούν στο εκάστοτε αυτόματο.

- AutomataVerse Simulator (https://www.automataverse.com/simulator/dfa)



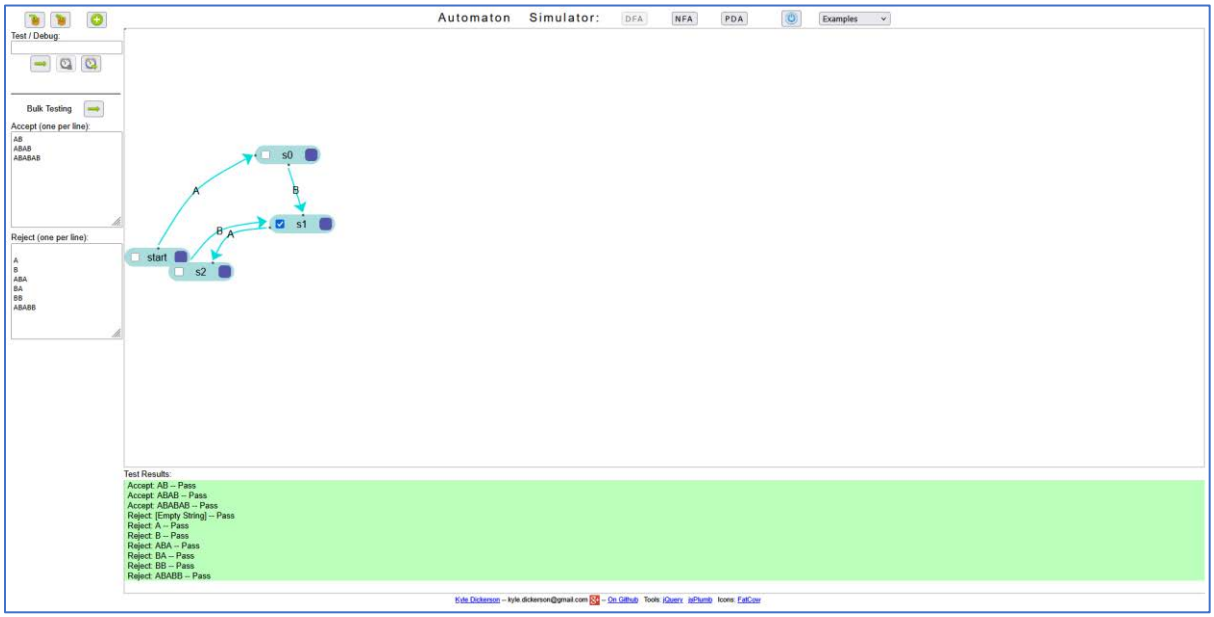
Εικόνα 2. Σχεδίαση DFA με χρήση του AutomataVerse

Ένα πιο σύγχρονο εργαλείο που επιτρέπει τη χειροκίνητη σχεδίαση ΝΠΑ μέσω προσθήκης καταστάσεων και μεταβάσεων. Ο χρήστης μπορεί να δημιουργήσει το δικό του αυτόματο, να το ελέγξει με το "Validate DFA" για να εντοπίσει σφάλματα και να το δοκιμάσει με εισαγωγή συμβολοσειρών μέσω της επιλογής "Validate". Επιπλέον, προσφέρει δυνατότητα εκτέλεσης βήμα-βήμα, για να μπορεί ο χρήστης να παρακολουθεί την πορεία της συμβολοσειράς. Με την επιλογή "Learn" του μενού εμφανίζονται προβλήματα τα οποία καλείται να επιλύσει ο χρήστης με σχεδίαση αυτομάτου που αναγνωρίζει δοθέντες συμβολοσειρές, καθιστώντας το ένα χρήσιμο εργαλείο, για εξάσκηση πάνω στα αυτόματα και γενικότερα στη Θεωρία Υπολογισμού.



Εικόνα 3. Στιγμιότυπο επίλυσης προβλήματος της εφαρμογής

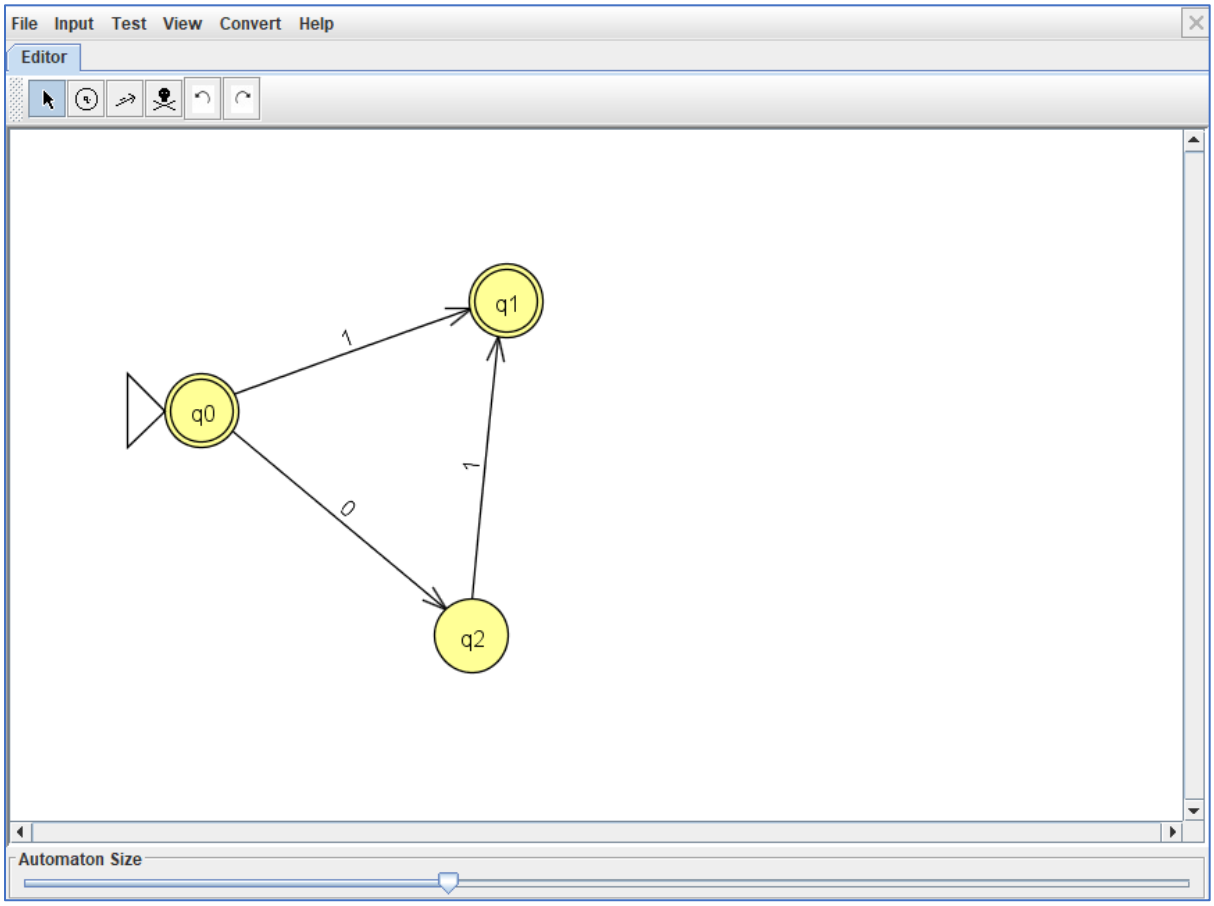
- Automaton Simulator (https://automatonsimulator.com/)



Εικόνα 4. Διεπαφή του Automaton Simulator με έτοιμο παράδειγμα αυτομάτου

Πρόκειται για μια εφαρμογή που επιτρέπει τη σχεδίαση DFA, NFA και PDA (PushDown Automaton). Ο χρήστης μπορεί να δημιουργήσει καταστάσεις και μεταβάσεις χειροκίνητα και να ορίσει αρχική και τελική κατάσταση. Παρέχει πεδίο για δοκιμή συμβολοσειράς (Test/Debug), αλλά και για μαζική δοκιμή (Bulk Testing) για πολλές συμβολοσειρές ταυτόχρονα. Τα αποτελέσματα εμφανίζονται στο κάτω μέρος της οθόνης και ορίζουν ποιες συμβολοσειρές γίνονται αποδεκτές και ποιες όχι.

- JFLAP (https://www.jflap.org/)



Εικόνα 5. Σχεδίαση αυτομάτου στο λογισμικό JFLAP

Το JFLAP (Java Formal Languages and Automata Package) είναι ένα ευρέως χρησιμοποιούμενο λογισμικό, το οποίο χρησιμοποιείται αρκετά σε πανεπιστημιακά μαθήματα, για τη σχεδίαση και προσομοίωση αυτομάτων αλλά και άλλων μοντέλων υπολογισμού. Υποστηρίζει DFA, NFA, PDA, Turing Machines, Γραμματικές, Κανονικές Εκφράσεις (ΚΕ), ακόμα και μετατροπές (από NFA σε DFA, από ΚΕ σε αυτόματα). Χρειάζεται εγκατάσταση διότι τρέχει σε Java και προσφέρει εκτενείς δυνατότητες, έχει όμως παρωχημένο και λιγότερο φιλικό περιβάλλον, συγκριτικά με τις υπόλοιπες εφαρμογές που αναπτύχθηκαν παραπάνω.

2.2 Σύγκριση Δυνατοτήτων

Η συγκριτική μελέτη των εργαλείων που παρουσιάστηκαν στην προηγούμενη ενότητα ανέδειξε σημαντικές διαφορές όσον αφορά τη λειτουργικότητα, τη φιλικότητα προς τον χρήστη, καθώς και την ευελιξία στη σχεδίαση και προσομοίωση

αυτομάτων. Παρακάτω συνοψίζονται τα βασικά χαρακτηριστικά και οι δυνατότητες κάθε εφαρμογής:

Εργαλείο	Υποστήριξη Αυτομάτων	Σχεδίαση από τον χρήστη	Προσομοίωση	Βηματική εκτέλεση	Επιπλέον δυνατότητες
DFA Simulator	Μόνο DFA	Όχι (μόνο έτοιμα παραδείγματα)	Ναι	Ναι	Εμφάνιση ΚΕ και γραμματικής
AutomataVerse Simulator	DFA και NFA σε διαφορετικό προσομοιωτή	Ναι	Ναι	Ναι	Demo και προβλήματα για επίλυση
Automaton Simulator	DFA, NFA, PDA	Ναι	Ναι	Ναι	Έτοιμα παραδείγματα
JFLAP	DFA, NFA, PDA, Turing Machines, ΚΕ, Γραμματικές	Ναι	Ναι	Ναι	Υποστηρίζει μετατροπές μεταξύ μοντέλων

Όπως φαίνεται στον πίνακα, το DFA Simulator απευθύνεται κυρίως σε χρήστες που θέλουν να κατανοήσουν βασικές έννοιες μέσω έτοιμων παραδειγμάτων, χωρίς τη δυνατότητα παρέμβασης στη δομή του αυτομάτου. Από την άλλη, τα AutomataVerse και Automaton Simulator προσφέρουν σημαντικά περισσότερη ελευθερία στον χρήστη, επιτρέποντας τη δημιουργία εξαρχής και τον πειραματισμό με νέες καταστάσεις και μεταβάσεις. Το Automaton Simulator ξεχωρίζει για την ευρύτερη υποστήριξη τύπων αυτομάτων, καθώς υποστηρίζει και PDAs, ενώ το AutomataVerse παρέχει έτοιμα προβλήματα που καλούν τον χρήστη για επίλυση με τον σχεδιασμό αυτομάτων. Τέλος, το JFLAP, παρότι λιγότερο μοντέρνο ως προς το περιβάλλον χρήσης, προσφέρει πληρέστερες δυνατότητες και θεωρείται το πιο ολοκληρωμένο εργαλείο, ειδικά για ακαδημαϊκή χρήση, λόγω της υποστήριξης μετατροπών μεταξύ διάφορων μοντέλων υπολογισμού.

Συνολικά, κάθε εργαλείο εξυπηρετεί διαφορετικούς στόχους: από την εκπαιδευτική κατανόηση εννοιών, μέχρι την πειραματική ανάπτυξη και βαθύτερη ανάλυση της θεωρίας υπολογισμού.

2.3 Ανάγκη για Νέα Εφαρμογή

Παρά την ύπαρξη αρκετών εργαλείων σχεδίασης και προσομοίωσης πεπερασμένων αυτομάτων διαπιστώνονται ορισμένοι περιορισμοί που καθιστούν

απαραίτητη την ανάπτυξη μιας νέας εφαρμογής που να ανταποκρίνεται καλύτερα στις ανάγκες των χρηστών.

Αρχικά, πολλά online εργαλεία δεν επιτρέπουν τον πλήρη έλεγχο του αυτομάτου από τον χρήστη, όπως τον ορισμό μεταβάσεων με το ίδιο σύμβολο από μια κατάσταση προς διαφορετικές καταστάσεις, την προσθήκη έψιλον-μεταβάσεων και την ευέλικτη διαχείριση των καταστάσεων. Ορισμένα δημιουργούν ΠΑ με αυτόματο τρόπο χωρίς να δίνουν στον χρήστη τη δυνατότητα σχεδίασης χειροκίνητα, περιορίζοντας τη διδακτική τους σημασία.

Επιπλέον, αρκετά εργαλεία δεν υποστηρίζουν ή έχουν περιορισμένες δυνατότητες όσον αφορά την αποθήκευση και φόρτωση αυτομάτων σε μορφή αρχείου (π.χ. JSON), ώστε να επεξεργάζονται αργότερα από τον χρήστη.

Ταυτόχρονα, λογισμικό όπως το JFLAP, παρότι ισχυρό, απαιτεί εγκατάσταση και περιβάλλον Java, κάτι που μπορεί να αποτελεί εμπόδιο σε συστήματα περιορισμένων δικαιωμάτων όπως εκπαιδευτικά περιβάλλοντα, όπου η εγκατάσταση λογισμικού δεν είναι εφικτή. Παράλληλα, το περιβάλλον του JFLAP θεωρείται ξεπερασμένο και λιγότερο φιλικό, ιδιαίτερα σε νέους χρήστες.

Τέλος, τα περισσότερα υπάρχοντα εργαλεία δεν υποστηρίζουν πολυγλωσσικό περιβάλλον, ενώ δεν παρέχουν οδηγίες χρήσης εντός της εφαρμογής, δημιουργώντας δυσκολίες σε μη έμπειρους χρήστες.

Η εφαρμογή που αναπτύχθηκε επιχειρεί να καλύψει αυτά τα κενά, προσφέροντας:

- Φιλικό και σύγχρονο περιβάλλον
- Υποστήριξη DFA , NFA αλλά και NFA-ε
- Αποθήκευση/φόρτωση τοπικά ή σε cloud server
- Πολυγλωσσική υποστήριξη
- Ενσωματωμένες οδηγίες χρήσης με αναδυόμενα παράθυρα
- Ευκολία χειρισμού για μαθητές και εκπαιδευτικούς

Η ανάπτυξη μιας τέτοιας εφαρμογής επιτρέπει τη διδακτική αξιοποίηση και την ευκολότερη πρόσβαση τόσο σε μαθητές, όσο και σε εκπαιδευτικούς, προσφέροντας έτσι ένα ανοιχτό, πρακτικό εργαλείο για κατανόηση, εξάσκηση και παρουσίαση της θεωρίας των αυτομάτων.

3. ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ

Η παρούσα πτυχιακή εργασία αναπτύχθηκε με τη χρήση σύγχρονων τεχνολογιών διαδικτύου, καθεμία από τις οποίες εξυπηρετεί διαφορετικές ανάγκες. Οι τεχνολογίες αυτές διακρίνονται σε δύο βασικές κατηγορίες: frontend και backend. Με τον όρο frontend αναφερόμαστε στο μέρος της εφαρμογής με το οποίο αλληλεπιδρά άμεσα ο χρήστης, περιλαμβάνει δηλαδή το γραφικό περιβάλλον και την παρουσίαση των δεδομένων. Αντίθετα, το backend περιλαμβάνει τη λογική του συστήματος, την επεξεργασία των δεδομένων και την επικοινωνία με τις βάσεις δεδομένων και τον διακομιστή.

3.1 HTML

Η HTML (HyperText Markup Language) αποτελεί τη βασική γλώσσα σήμανσης που χρησιμοποιείται για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Σχεδιάστηκε για τη δόμηση και την οργάνωση περιεχομένου στο διαδίκτυο, όπως κειμένου, εικόνας, συνδέσμων κ.λπ. Γνώρισε μεγάλη άνθιση την δεκαετία του 1990, ενώ μέχρι σήμερα έχουν αναπτυχθεί διάφορες εκδόσεις της (HTML 5.0, XHTML, DHTML), οι οποίες εισάγουν νέα στοιχεία όπως <video>, <audio>, <section> ή επαναπροσδιορίζουν τα ήδη υπάρχοντα.

Η HTML αποτελείται από στοιχεία (elements) τα οποία και ορίζουν το βασικό συστατικό της δομής μιας ιστοσελίδας. Τέτοια στοιχεία είναι οι επικεφαλίδες, οι παράγραφοι, οι λίστες και οι πίνακες που επιτρέπουν την οργάνωση και παρουσίαση της πληροφορίας με σαφή και δομημένο τρόπο. Τα στοιχεία αυτά περικλείονται σε ετικέτες (tags), οι οποίες είναι συνήθως σε ζευγάρια, για να οριστεί με αυτόν τον τρόπο η αρχή και το τέλος μιας εντολής.

Κάθε αρχείο HTML αποτελείται από ορισμένες καθιερωμένες ετικέτες, όπως αυτές για το κείμενο επικεφαλίδας και το κείμενο σώματος (head, body).

Εικόνα 6. Βασική δομή αρχείου HTML με ετικέτες head και body

Η HTML μπορεί να δίνει τη δομή σε μια ιστοσελίδα, αλλά για να επιτευχθεί η μορφοποίησή της χρησιμοποιείται η CSS, η οποία συνεργάζεται άμεσα με την HTML και είναι υπεύθυνη για την αισθητική βελτίωση και το σχεδιαστικό κομμάτι, όπως τα χρώματα, τις γραμματοσειρές, τις αποστάσεις.

3.2 CSS

Η CSS (Cascading Style Sheets) αποτελεί τη γλώσσα που χρησιμοποιείται για την αισθητική διαμόρφωση των ιστοσελίδων. Είναι υπεύθυνη για την παρουσίαση, όπως τα χρώματα, τις γραμματοσειρές, τα περιθώρια, τη στοίχιση, τις διαστάσεις και όλα τα οπτικά χαρακτηριστικά της σελίδας. Μέσω της CSS, επιτυγχάνεται ο διαχωρισμός περιεχομένου και μορφοποίησης, γεγονός που διευκολύνει τη συντήρηση και την επαναχρησιμοποίηση του κώδικα, ειδικά σε έργα μεγάλης κλίμακας.

Η CSS εφαρμόζεται με τρεις βασικούς τρόπους: *Inline* (μέσα στο ίδιο το στοιχείο HTML, π.χ. `<p style=”color:red;”>`), *Internal* (εντός του <style> μέσα στο <head> του αρχείου HTML) και *External* (μέσω ξεχωριστού αρχείου .css, το οποίο και εισάγεται με `<link rel="stylesheet” href=”file.css”>`). Από πλευράς προτεραιότητας, τα inline στυλ υπερισχύουν των υπολοίπων, ενώ όταν υπάρχουν αντικρουόμενοι κανόνες, η CSS ακολουθεί την ιεραρχία, γνωστή και ως cascading, όπου λαμβάνονται υπόψη η σειρά εμφάνισης, η ειδικότητα των selectors και η σημασία τους.

Οι selectors είναι βασικό στοιχείο της CSS, καθώς επιτρέπουν την εφαρμογή κανόνων σε συγκεκριμένα στοιχεία ή και ομάδες στοιχείων. Υπάρχουν απλοί selectors, όπως το όνομα ενός στοιχείου (p, div), κλάσεις (.class) και αναγνωριστικά (#id), καθώς και πιο σύνθετοι που επιτρέπουν τον συνδυασμό συγκεκριμένων χαρακτηριστικών μέσα στο DOM (Document Object Model).

Η CSS έχει εξελιχθεί με την πάροδο του χρόνου και ειδικά με τις τελευταίες εκδόσεις (CSS3), έχει αποκτήσει ισχυρότερες δυνατότητες όπως animations, media queries, layouts κλπ.

Η χρήση της CSS δεν προοδιορίζεται μόνο στην αισθητική, αλλά σχετίζεται άμεσα με την εμπειρία του τελικού χρήστη και τη διαδραστικότητα. Ένας καλά δομημένος κώδικας CSS, συμβάλει σε μια λειτουργική και προσβάσιμη εμπειρία πλοήγησης, κάνοντας τον συνδυασμό HTML και CSS τη θεμελιώδη βάση κάθε σύγχρονης ιστοσελίδας.

3.3 JavaScript

Η JavaScript είναι μία από τις βασικότερες γλώσσες προγραμματισμού για την ανάπτυξη εφαρμογών στο διαδίκτυο και χρησιμοποιείται σχεδόν σε κάθε σύγχρονη ιστοσελίδα. Πρόκειται για μία γλώσσα με αντικειμενοστραφή φιλοσοφία (object oriented), αλλά και υποστήριξη ασύγχρονου προγραμματισμού (callbacks, async κ.λπ.) γεγονός που την καθιστά εξαιρετικά ευέλικτη και αποτελεσματική για την αλληλεπίδραση του χρήστη με την εφαρμογή σε πραγματικό χρόνο.

Αρχικά, είχε σχεδιαστεί για να χρησιμοποιείται στο client-side (στην πλευρά του χρήστη) κυρίως για την προσθήκη διαδραστικότητας στις ιστοσελίδες, όπως η διαχείριση κουμπιών, μενού κ.ά. Με την πάροδο του χρόνου και την ανάπτυξη εργαλείων όπως το Node.js, η χρήση της επεκτάθηκε και στο server-side. Αυτό άνοιξε τον δρόμο για την ανάπτυξη πλήρως λειτουργικών εφαρμογών, με κοινή γλώσσα προγραμματισμού τόσο στο frontend όσο και στο backend, διευκολύνοντας έτσι τη συντήρηση του κώδικα και επιταχύνοντας τη διαδικασία ανάπτυξης.

Η JavaScript αποτελεί τη βάση για πολλά σύγχρονα frameworks και βιβλιοθήκες του frontend development. Ανάμεσα τους ξεχωρίζουν το React, το Vue.js και το Angular, τα οποία παρέχουν δομές και αρχιτεκτονικές για την

ανάπτυξη πολύπλοκων, επεκτάσιμων και αποδοτικών user interfaces, καθώς και εργαλεία όπως το Express.js στο backend, τα οποία συνθέτουν συνολικά, ένα ολοκληρωμένο οικοσύστημα ανάπτυξης εφαρμογών.

Τέλος, η ευρεία υποστήριξη της γλώσσας από όλους τους σύγχρονους φυλλομετρητές (browsers), σε συνδυασμό με την διαρκή εξέλιξη της, εξασφαλίζουν ότι θα παραμείνει στην πρώτη γραμμή της τεχνολογίας και θα συνεχίσει να αποτελεί αναπόσπαστο εργαλείο για την ανάπτυξη εφαρμογών στο διαδίκτυο.

3.4 Node και Ανάπτυξη Backend

Η ανάπτυξη του backend αποτελεί βασικό στοιχείο σε κάθε διαδικτυακή εφαρμογή, καθώς περιλαμβάνει την υλοποίηση της λογικής της εφαρμογής, την αλληλεπίδραση με τη βάση δεδομένων και τη διαχείριση αιτημάτων από και προς τον πελάτη (frontend).

Μία από τις πιο σύγχρονες και δημοφιλείς τεχνολογίες για την υλοποίηση backend είναι το Node.js. Πρόκειται για ένα περιβάλλον εκτέλεσης JavaScript στον διακομιστή, το οποίο βασίζεται στη μηχανή V8 της Google. Χάρη στην ασύγχρονη αρχιτεκτονική του, προσφέρει υψηλές επιδόσεις και είναι ιδανικό για εφαρμογές με έντονη αλληλεπίδραση, πολλαπλά ταυτόχρονα I/O αιτήματα και συνεχή ροή δεδομένων, όπως εφαρμογές με chat ή real-time dashboards.

Συχνά, το Node.js συνδυάζεται με το Express, ένα ευέλικτο και ελαφρύ framework που απλοποιεί τη δημιουργία web εφαρμογών και RESTful APIs, και επιτρέπει παράλληλα την εύκολη διαχείριση HTTP routes.

Η χρήση Node.js και Express έχει καθιερωθεί στην ανάπτυξη σύγχρονων διαδικτυακών εφαρμογών λόγω της ταχύτητας ανάπτυξης, της επεκτασιμότητας και της ενιαίας χρήσης JavaScript σε όλο το φάσμα της εφαρμογής.

Εκτός από το Node.js, μια ακόμη ιδιαίτερα διαδεδομένη τεχνολογία backend, είναι η PHP, η οποία χρησιμοποιείται ευρέως εδώ και δεκαετίες για τη δημιουργία δυναμικών ιστοσελίδων. Αν και θεωρείται παλαιότερη τεχνολογία σε σχέση με το Node.js, παραμένει εξαιρετικά δημοφιλής, κυρίως λόγω της απλότητας και της συμβατότητας της με άλλα συστήματα. Η PHP είναι κατάλληλη για web εφαρμογές

και συνεχίζει να υποστηρίζεται ενεργά με σύγχρονες βελτιώσεις και frameworks, όπως το Laravel, που προσφέρουν δομή και ευκολία στην ανάπτυξη.

Επιπλέον, άλλες γλώσσες και τεχνολογίες, όπως η Python με τα frameworks Django και Flask, η Ruby με το Ruby on Rails, προσφέρουν εξαιρετικές δυνατότητες για την ανάπτυξη backend εφαρμογών, δίνοντας έμφαση στην καθαρή σύνταξη και την ευκολία συντήρησης του κώδικα. Η επιλογή τεχνολογίας εξαρτάται κάθε φορά από τις ανάγκες του έργου αλλά και τη φιλοσοφία σχεδίασης της εφαρμογής.

3.5 Βάσεις Δεδομένων

Οι βάσεις δεδομένων (databases) αποτελούν αναπόσπαστο μέρος κάθε διαδικτυακής εφαρμογής, καθώς επιτρέπουν την αποθήκευση, διαχείριση και ανάκτηση δεδομένων με αποδοτικό τρόπο. Σε αυτήν την πτυχιακή εργασία, χρησιμοποιήθηκε η PostgreSQL, μια ισχυρή βάση δεδομένων. Ωστόσο, υπάρχουν πολλές άλλες βάσεις δεδομένων που χρησιμοποιούνται στον τομέα της ανάπτυξης εφαρμογών, ανάλογα με τις ανάγκες της κάθε εφαρμογής.

1. PostgreSQL

   Η PostgreSQL είναι μια σχεσιακή βάση δεδομένων ανοιχτού κώδικα που υποστηρίζει το πρότυπο SQL (Structure Query Language). Προσφέρει υψηλή αξιοπιστία, σταθερότητα σε σύνθετες εφαρμογές και επεκτασιμότητα. Υποστηρίζει επίσης σύνθετους τύπους δεδομένων, indexing, triggers και άλλες προηγμένες δυνατότητες.

2. MySQL

   Η MySQL είναι μια από τις πιο δημοφιλείς βάσεις δεδομένων και χρησιμοποιείται ευρέως για διαδικτυακές εφαρμογές και ιστοσελίδες. Είναι γνωστή για την απλότητα και την αποτελεσματικότητα της, καθώς και για τη μεγάλη κοινότητα χρηστών και υποστήριξης. Υποστηρίζει και αυτή το πρότυπο SQL και παρέχει επεκτασιμότητα, ευχρηστία και ευκολία διαχείρισης, καθιστώντας την κατάλληλη για εφαρμογές μικρού και μεσαίου μεγέθους.

3. MongoDB

   Η MongoDB είναι μία βάση δεδομένων, που αποθηκεύει τα δεδομένα σε έγγραφα τύπου JSON αντί για παραδοσιακούς πίνακες. Είναι κατάλληλη για εφαρμογές που απαιτούν ευελιξία στις δομές δεδομένων και δυνατότητα ταχείας κλιμάκωσης. Χρησιμοποιείται κυρίως σε περιβάλλοντα με δεδομένα που εξελίσσονται συνεχώς ή απαιτούν ταχύτερη αναβάθμιση και προσαρμογή.

4. SQLite

   Αυτήν είναι μια ελαφριά βάση δεδομένων που δεν απαιτεί εξωτερικό διακομιστή για τις λειτουργίες της. Χρησιμοποιείται συχνά σε εφαρμογές κινητών, desktop λογισμικό και ελαφριές web εφαρμογές που απαιτούν μια απλή και γρήγορη λύση ,όσον αφορά την αποθήκευση δεδομένων.

5. Microsoft SQL Server

   Η Microsoft SQL Server είναι ένα εμπορικό RDBMS (Relational Database Management System) που αναπτύσσεται από τη Microsoft. Παρέχει πλήρη υποστήριξη για εφαρμογές επιχειρηματικής κλίμακας και είναι ενσωματωμένο στο οικοσύστημα των τεχνολογιών της Microsoft.

Συμπερασματικά, η επιλογή της βάσης δεδομένων εξαρτάται σε μεγάλο βαθμό από τις ανάγκες και τις απαιτήσεις της εφαρμογής, όπως ο τύπος δεδομένων, η ανάγκη για ταχύτητα και επεκτασιμότητα. Οι βάσεις που αναφέρθηκαν αντιπροσωπεύουν μερικές από τις δημοφιλείς και αξιόπιστες λύσεις στον χώρο της ανάπτυξης λογισμικού.

4. ΠΕΡΙΓΡΑΦΗ, ΑΡΧΙΤΕΚΤΟΝΙΚΗ & ΥΛΟΠΟΙΗΣΗ

Στο προηγούμενο κεφάλαιο, αναπτύχθηκαν γνωστές τεχνολογίες διαδικτύου για την ανάπτυξη ιστοσελίδων. Σε αυτό το κεφάλαιο, θα γίνει μια λεπτομερής ανάλυση για τις τεχνολογίες που αξιοποιήθηκαν για την υλοποίηση της διαδικτυακής εφαρμογής, καθώς και τις θεωρητικές έννοιες που χρησιμοποιήθηκαν και είναι αναγκαίες να γνωρίζει ο χρήστης για τη σωστή και ομαλή χρήση της εφαρμογής.

4.1 Θεωρητικό Υπόβαθρο

Τα Πεπερασμένα Αυτόματα (ΠΑ) γνωστά και ως Μηχανές Πεπερασμένων Καταστάσεων (Finite State Machines) είναι μαθηματικά υπολογιστικά μοντέλα που χρησιμοποιούνται για την αναγνώριση όλων των συμβολοσειρών μιας γλώσσας. Είναι δηλαδή μια μηχανή αναγνώρισης, η οποία δέχεται ως είσοδο μία συμβολοσειρά και παράγει ως έξοδο ένα αποτέλεσμα. Αυτό το αποτέλεσμα μπορεί να είναι είτε αποδοχή είτε απόρριψη. Το αυτόματο ξεκινάει από την αρχική του κατάσταση και διαβάζει την συμβολοσειρά ανά χαρακτήρα και ανάλογα με την τρέχουσα κατάσταση που βρίσκεται, επιλέγει την νέα κατάσταση που θα μεταβεί, με βάση αυτόν τον χαρακτήρα. Το αποτέλεσμα είναι αποδοχή, όταν η μηχανή διαβάσει όλους τους χαρακτήρες της συμβολοσειράς και έχει καταλήξει σε μία τελική κατάσταση ή αλλιώς κατάσταση αποδοχής. Αντίθετα εάν δεν βρίσκεται σε τέτοια κατάσταση και δεν υπάρχουν άλλοι χαρακτήρες να διαβάσει από την συμβολοσειρά, το αποτέλεσμα που παράγει είναι απόρριψη.

Γενικά, ένα αυτόματο ορίζεται ως μια πλειάδα :

$$M = \{Q, \Sigma, q, F, \delta\}$$

όπου:

- Q: το σύνολο των καταστάσεων, δηλαδή όλες οι δυνατές καταστάσεις στις οποίες μπορεί να βρεθεί το αυτόματο κατά την δοκιμή μιας συμβολοσειράς.
- Σ: το σύνολο των συμβόλων εισόδου, γνωστό και ως αλφάβητο, που περιέχει όλα τα επιτρεπτά σύμβολα από τα οποία αποτελούνται οι συμβολοσειρές που δέχεται το αυτόματο.
- q: η αρχική κατάσταση, δηλαδή η κατάσταση από την οποία ξεκινά ο έλεγχος κάθε συμβολοσειράς.
- F: το σύνολο των τελικών καταστάσεων (ή καταστάσεων αποδοχής), στις οποίες πρέπει να καταλήξει το αυτόματο για να θεωρηθεί μια συμβολοσειρά αποδεκτή.
- δ: η συνάρτηση μετάβασης, που ορίζει για κάθε συνδυασμό κατάστασης και συμβόλου εισόδου σε ποια κατάσταση θα μεταβεί το αυτόματο.

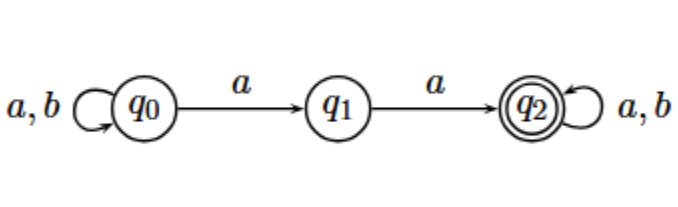
Με βάση τα παραπάνω στοιχεία, ένα ΠΑ καθορίζει πώς επεξεργάζεται κάθε σύμβολο εισόδου και ποια πορεία ακολουθεί, μέχρι να αποδεχθεί ή να απορρίψει τη συμβολοσειρά.

Τα πεπερασμένα αυτόματα διακρίνονται σε Μη-ντετερμινιστικά (ΜΠΑ) και Ντετερμινιστικά (ΝΠΑ).

4.1.1 Μη-ντετερμινιστικά ΠΑ

Ένα μη-ντετερμινιστικό ΠΑ (Non-deterministic Finite Automaton - NFA) είναι ένα μοντέλο το οποίο επιτρέπει πολλαπλές εναλλακτικές μεταβάσεις από μία κατάσταση για το ίδιο σύμβολο εισόδου. Σε ένα NFA είναι δυνατό να υπάρχουν περισσότερες από μία μεταβάσεις από μία κατάσταση για το ίδιο σύμβολο ή και καταστάσεις χωρίς καθορισμένες μεταβάσεις για όλα τα σύμβολα. Η αποδοχή μιας συμβολοσειράς συμβαίνει αν υπάρχει έστω μία διαδρομή μέσω του αυτομάτου, που οδηγεί σε τελική κατάσταση αφού διαβαστεί ολόκληρη η συμβολοσειρά.

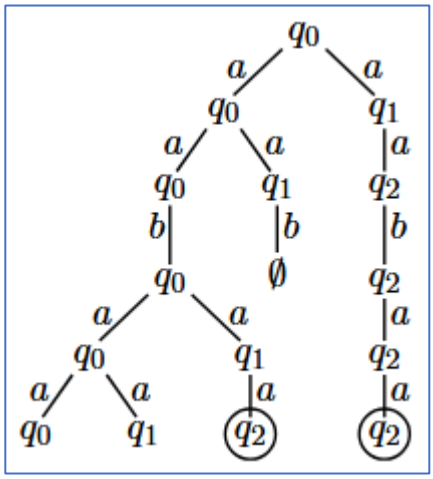
Επιπλέον, ένα ιδιαίτερο είδος μη-ντετερμινιστικού ΠΑ είναι το NFA-ε (epsilon-NFA), στο οποίο επιτρέπονται μεταβάσεις χωρίς να καταναλώνεται κάποιο σύμβολο εισόδου, γνωστές και ως έψιλον-μεταβάσεις. Αυτές οι μεταβάσεις συμβολίζονται με το γράμμα ε (δηλαδή κενό σύμβολο) και επιτρέπουν στο αυτόματο να αλλάξει κατάσταση, χωρίς να διαβάσει κάποιο σύμβολο σαν είσοδο. Το NFA-ε χρησιμοποιείται συχνά στη μεταγλώττιση και στη θεωρία κανονικών εκφράσεων, καθώς διευκολύνει τη δημιουργία αυτομάτων από σύνθετες εκφράσεις. Θεωρητικά αποδεικνύεται ότι κάθε NFA-ε μπορεί να μετατραπεί σε ισοδύναμο NFA και αυτό με τη σειρά του σε DFA. Επίσης ένα NFA μπορεί να αναπαρασταθεί και από έναν πίνακα καταστάσεων όπως παρακάτω:



	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\emptyset$
(q_2)	$\{q_2\}$	$\{q_2\}$

Εικόνα 7. NFA για $\boldsymbol{L} := \{\boldsymbol{w} \in \boldsymbol{\Sigma}* \mid \boldsymbol{w}$ **περιέχει δύο συνεχόμενα** $\boldsymbol{\alpha}\}$

Η ακολουθία των καταστάσεων σε ένα NFA δεν είναι γραμμική, και γι' αυτό μπορεί να θεωρηθεί ως ένα δένδρο. Παρακάτω είναι το αντίστοιχο δένδρο υπολογισμού για το παράδειγμα 4.1, για είσοδο "aabaa" και όπως φαίνεται η συμβολοσειρά είναι αποδεκτή, καθώς υπάρχει τουλάχιστον μία διαδρομή που να την αποδέχεται.



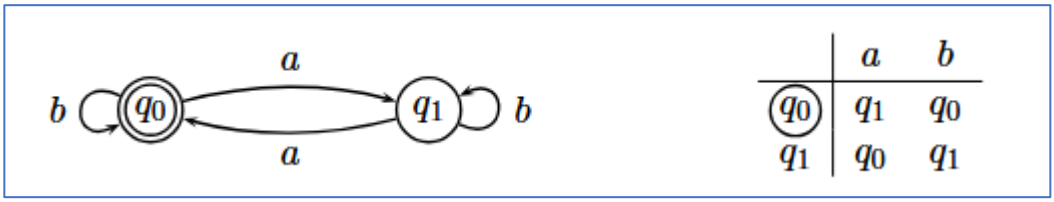
Εικόνα 8. Δένδρο υπολογισμού για το NFA της εικόνας 7

4.1.2 Ντετερμινιστικά ΠΑ

Ένα ντετερμινιστικό ΠΑ (Deterministic Finite Automaton - DFA) είναι μια ειδική περίπτωση ενός NFA, όπου για κάθε κατάσταση και για κάθε σύμβολο εισόδου, υπάρχει ακριβώς μία μετάβαση. Επίσης, σε αυτό το είδος αυτομάτου δεν επιτρέπονται μεταβάσεις με κενό σύμβολο (ε). Αυτό σημαίνει ότι η εκτέλεση του DFA είναι πλήρως καθορισμένη και δεν εμπεριέχει καμία μορφή αβεβαιότητας ή εναλλακτικών διαδρομών. Τυπικά, σε ένα DFA ισχύει η εξής ιδιότητα:

Για κάθε κατάσταση $q \in Q$ και για κάθε σύμβολο $\alpha \in \Sigma$, υπάρχει ακριβώς μία μετάβαση της μορφής $\delta(q,\alpha)$ που οδηγεί σε επόμενη κατάσταση.

Με άλλα λόγια, ένα DFA έχει μοναδική ενεργή διαδρομή ανά πάσα στιγμή και χαρακτηρίζεται από πλήρη καθοριστικότητα. Η προβλεψιμότητα αυτή τα καθιστά ιδανικά για υλοποιήσεις όπου απαιτείται γραμμική επεξεργασία εισόδου.



Εικόνα 9. DFA για $\boldsymbol{L = \{w \in \Sigma^* \mid w \text{ περιέχει άρτιο πλήθος από } \alpha\}}$

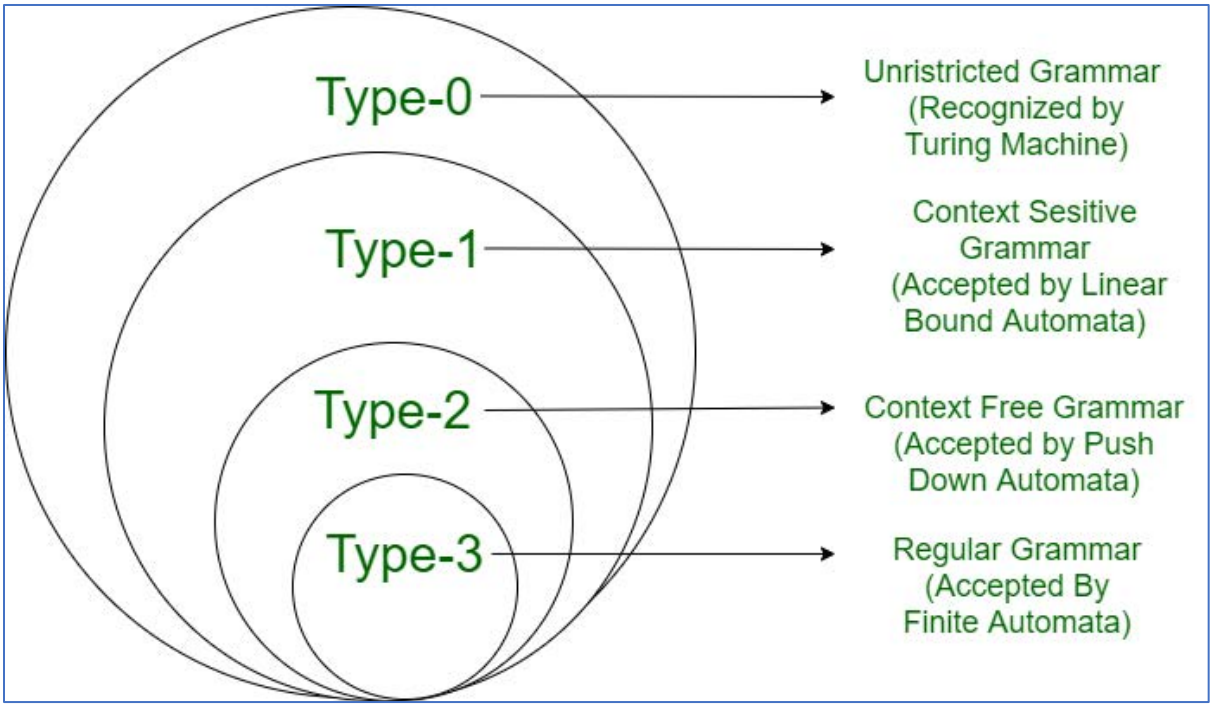
Η ακολουθία των καταστάσεων σε ένα DFA είναι γραμμική και μονοσήμαντη, σε αντίθεση με τα NFA όπου μπορεί να αναπτυχθεί δένδρο πιθανών διαδρομών. Αυτό καθιστά τα DFA ιδιαίτερα αποτελεσματικά σε εφαρμογές ανάλυσης και αναγνώρισης συμβολοσειρών.

4.1.3 Τυπικές Γλώσσες

Οι φυσικές γλώσσες, όπως η Ελληνική, χαρακτηρίζονται από ασάφεια και εξάρτηση από τα συμφραζόμενα. Μια χαρακτηριστική περίπτωση είναι η πρόταση: «Ο Γιάννης προχώρησε προς τη γυναίκα με την ανθοδέσμη», η οποία είναι διφορούμενη ως προς το ποιος κρατούσε την ανθοδέσμη [1]. Αντίθετα, οι γλώσσες προγραμματισμού αποτελούν τυπικές γλώσσες και είναι σχεδιασμένες ώστε να μην επιδέχονται διφορούμενη ερμηνεία, ακολουθώντας αυστηρά και σαφώς καθορισμένους συντακτικούς κανόνες.

Μια τυπική γλώσσα (formal language) ορίζεται ως ένα (ενδεχομένως άπειρο) σύνολο από πεπερασμένου μήκους συμβολοσειρές, οι οποίες κατασκευάζονται από ένα συγκεκριμένο αλφάβητο συμβόλων.

Οι τυπικές γλώσσες και οι γραμματικές τους κατατάσσονται στην ιεραρχία Chomsky, η οποία περιλαμβάνει τέσσερις κατηγορίες: τύπου 0 (χωρίς περιορισμούς), τύπου 1 (με συμφραζόμενα), τύπου 2 (χωρίς συμφραζόμενα) και τύπου 3 (κανονικές γραμματικές). Κάθε κατηγορία είναι υποσύνολο της προηγούμενης. Στην πράξη, οι κανονικές και οι χωρίς συμφραζόμενα γραμματικές είναι οι πιο συχνά χρησιμοποιούμενες στην Ανάλυση Γλωσσών και στους Μεταγλωττιστές.



Εικόνα 10. Ιεραρχία Chomsky στη Θεωρία Υπολογισμού

4.1.4 Κανονικές Εκφράσεις

Οι κανονικές εκφράσεις (ΚΕ ή Regular Expressions) αποτελούν τυπικό τρόπο περιγραφής συνόλων συμβολοσειρών και συνδέονται άμεσα με τις κανονικές γλώσσες και τα πεπερασμένα αυτόματα. Χρησιμοποιούν σύμβολα της αλφαβήτου, μαζί με ειδικούς τελεστές, για να εκφράσουν γλωσσικά πρότυπα με συνοπτικό και τυπικά καθορισμένο τρόπο. Για παράδειγμα, η έκφραση b* περιγράφει όλες τις συμβολοσειρές που αποτελούνται από μηδέν ή περισσότερες εμφανίσεις του συμβόλου b.

Οι βασικοί ειδικοί τελεστές στις κανονικές εκφράσεις περιλαμβάνουν:

- Το σύμβολο ***** (Kleene star), το οποίο δηλώνει μηδέν ή περισσότερες εμφανίσεις του προηγούμενου χαρακτήρα ή ομάδας. Για παράδειγμα, η έκφραση a* αποδέχεται την κενή συμβολοσειρά, την “a”, την “aa”, την “aaa” κ.ο.κ.
- Το σύμβολο **+**, το οποίο δηλώνει μία ή περισσότερες εμφανίσεις, με άλλα λόγια τουλάχιστον μία. Η έκφραση a+ αποδέχεται τις συμβολοσειρές “a”, “aa”, “aaa”, αλλά όχι την κενή συμβολοσειρά.

- Το σύμβολο **?** δηλώνει ότι το προηγούμενο στοιχείο είναι προαιρετικό, δηλαδή εμφανίζεται μία ή καμία φορά. Για παράδειγμα, η a?b αποδέχεται τόσο το "b" όσο και το "ab", αλλά όχι το "aab".

Για παράδειγμα, έχουμε την ΚΕ: $R = a(b|c)*$

Η παραπάνω ΚΕ περιγράφει τις συμβολοσειρές που ξεκινούν με το σύμβολο a και ακολουθούνται από οποιοδήποτε πλήθος (ακόμη και μηδέν) συμβόλων b ή c. Συνεπώς ανήκουν στη γλώσσα λέξεις όπως "a", "ab", "ac", "abcb", "acbc" κ.ά.

Οι ΚΕ είναι θεωρητικά ισοδύναμες με τα ΠΑ: κάθε κανονική έκφραση μπορεί να μετατραπεί σε ισοδύναμο NFA και το αντίστροφο. Η ευρεία χρήση τους σε πρακτικά εργαλεία αποδεικνύει τη χρησιμότητα τους και πέραν της θεωρητικής βάσης.

4.1.5 Κανονικές Γραμματικές

Οι Κανονικές Γραμματικές (Regular Grammars) αποτελούν το απλούστερο είδος γραμματικών στην ιεραρχία Chomsky και αντιστοιχούν στις κανονικές γλώσσες και στα πεπερασμένα αυτόματα. Οι γραμματικές αυτού του τύπου περιλαμβάνουν αυστηρά καθορισμένους κανόνες παραγωγής, επιτρέποντας τη δημιουργία γλωσσών που δεν απαιτούν συμφραζόμενα ή σύνθετη συντακτική δομή.

Μια κανονική γραμματική G ορίζεται ως G = {T, N, P, S} όπου:

- T είναι το σύνολο των τερματικών συμβόλων [Terminal Symbol]
- N είναι το σύνολο των μη τερματικών συμβόλων [non-Terminal Symbol]
- P είναι το σύνολο των κανόνων παραγωγής [Production Rules]
- S είναι το αρχικό σύμβολο [Start Symbol]

Οι κανόνες παραγωγής έχουν τη μορφή:

- A → aB (τερματικό και μη τερματικό)
- A → a (μόνο τερματικό)
- A → ε (για την παραγωγή κενής συμβολοσειράς)

Για παράδειγμα, έχουμε την γραμματική G:

Μη τερματικά: $N = \{S\}$

Τερματικά: $T = \{a, b\}$

Κανόνες:

- $S \rightarrow aS$
- $S \rightarrow bS$
- $S \rightarrow \varepsilon$

Αυτή η κανονική γραμματική παράγει τη γλώσσα όλων των συμβολοσειρών που αποτελούνται από τα σύμβολα a και b, σε οποιοδήποτε πλήθος και σειρά, συμπεριλαμβανομένης και της κενής συμβολοσειράς.

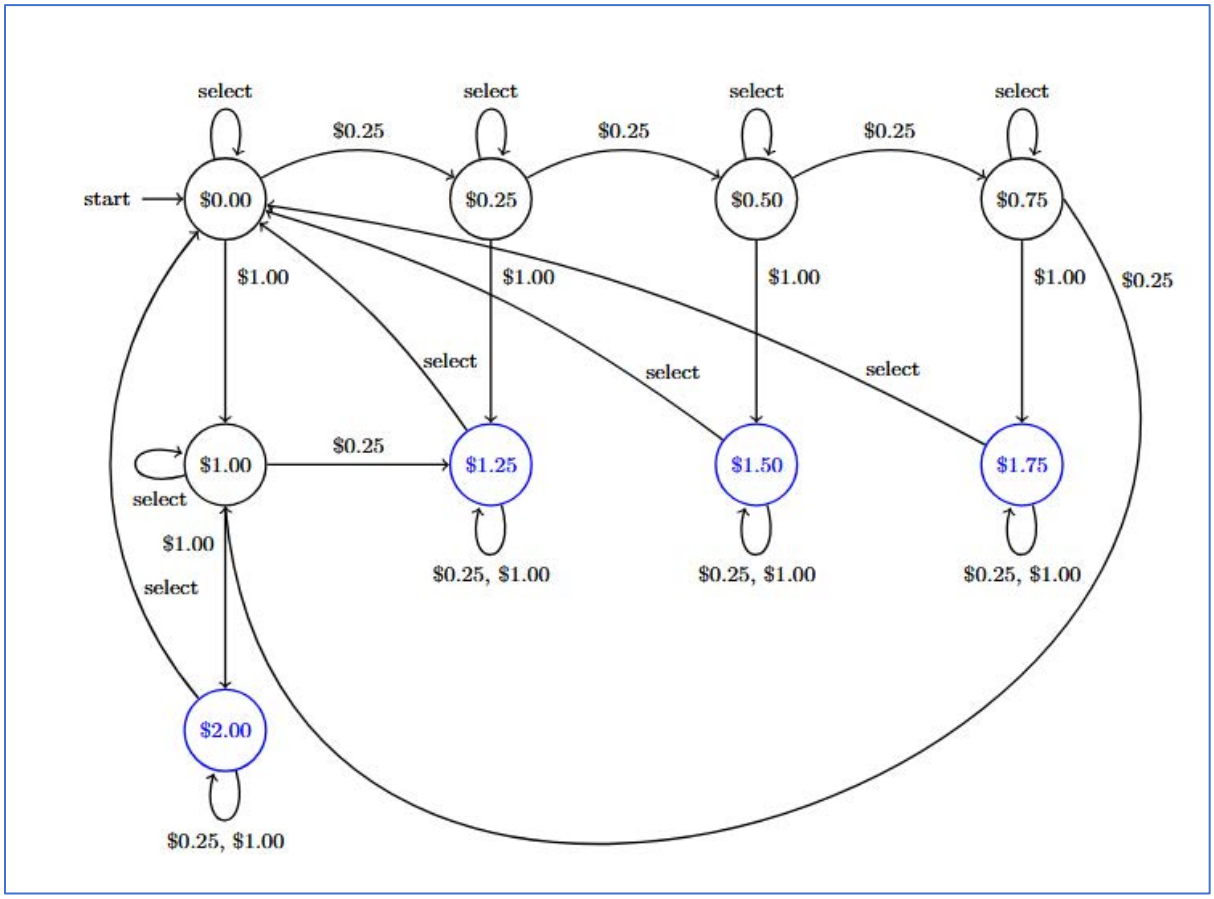
Η παραγόμενη γλώσσα είναι: L(G) = {ε, a, b, ab, ba, aa, bb, aab, …} η οποία είναι κανονική και μπορεί να αναπαρασταθεί είτε με κανονική έκφραση: (a | b)*, είτε με πεπερασμένο αυτόματο.

4.1.6 Χρήσεις ΠΑ

Τα ΠΑ χρησιμοποιούνται σε διάφορους τομείς της Επιστήμης των Υπολογιστών, όπως στην σχεδίαση μεταγλωττιστών για την αναγνώριση κανονικών εκφράσεων, σε εργαλεία επαλήθευσης λογισμικού, στη μοντελοποίηση πρωτοκόλλων επικοινωνίας, καθώς και σε συστήματα αναγνώρισης προτύπων. Επιπλέον βρίσκουν εφαρμογή στην ανάλυση κειμένου, στη σχεδίαση των μηχανών αναζήτησης, αλλά και σε απλές λογικές διεργασίες όπου χρειάζεται να παρακολουθείται η εξέλιξη καταστάσεων με βάση ακολουθίες «συμβάντων».

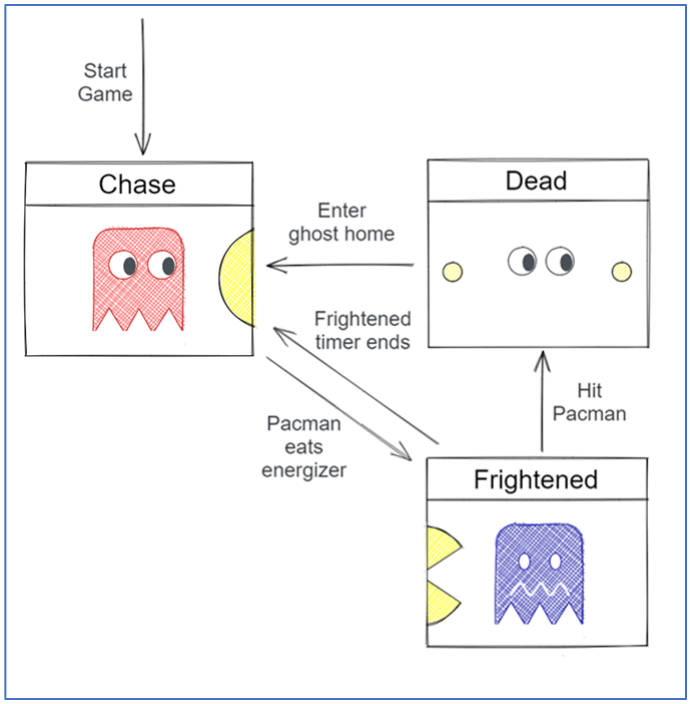
Ένα χαρακτηριστικό παράδειγμα χρήσης ΠΑ στην καθημερινότητα είναι η λειτουργία ενός αυτόματου πωλητή (vending machine). Στην περίπτωση αυτή, το κάθε στάδιο εισαγωγής χρημάτων αντιστοιχεί σε μια κατάσταση του αυτομάτου, ενώ η εισαγωγή ενός νομίσματος αποτελεί το σύμβολο εισόδου που προκαλεί μετάβαση σε άλλη κατάσταση, αυξάνοντας το υπόλοιπο. Μόλις συγκεντρωθεί το απαιτούμενο ποσό, το αυτόματο μεταβαίνει σε μια τελική κατάσταση αποδοχής, όπου εκτελείται η παραλαβή του προϊόντος. Αντίστοιχα, η επιστροφή χρημάτων ή η ακύρωση μπορεί να αναπαραχθεί με μετάβαση σε άλλη κατάσταση. Ένα τέτοιο μοντέλο επιτρέπει την

ανάλυση και τον σχεδιασμό της λογικής λειτουργίας ενός αυτόματου συστήματος με ακρίβεια.



Εικόνα 11. Παράδειγμα DFA που περιγράφει τη συμπεριφορά ενός αυτόματου πωλητή

Εκτός από τις εφαρμογές τους στον τομέα της Πληροφορικής, τα ΠΑ βρίσκουν χρήση και στη βιομηχανία των βιντεοπαιχνιδιών, ιδιαίτερα στον σχεδιασμό της τεχνητής νοημοσύνης (AI) των NPCs (Non Playable Characters). Ένα χαρακτηριστικό παράδειγμα είναι το κλασικό παιχνίδι Pac-Man, στο οποίο η συμπεριφορά των τεσσάρων φαντασμάτων καθορίζεται από ένα FSM (Finite State Machine). Κάθε φάντασμα μπορεί να βρίσκεται σε μία από τις εξής καταστάσεις: Chase, Dead και Frightened. Η μετάβαση από τη μία κατάσταση στην άλλη εξαρτάται είτε από το πέρασμα του χρόνου, είτε από την κατανάλωση energizer. Η χρήση ενός ΠΑ για τον έλεγχο της συμπεριφοράς των φαντασμάτων επιτρέπει ένα προβλέψιμο, αλλά και ενδιαφέρον μοτίβο κινήσεων, συμβάλλοντας στη δημιουργία στρατηγικής από τον παίκτη.



Εικόνα 12. Παράδειγμα ΠΑ για τον έλεγχο συμπεριφοράς φαντασμάτων στο Pac-Man

4.2 Σχεδίαση Συστημάτων

Η σχεδίαση των συστημάτων αποτελεί καθοριστικό στάδιο στην ανάπτυξη μιας διαδικτυακής εφαρμογής, καθώς επηρεάζει τόσο τη λειτουργικότητα όσο και την ευκολία συντήρησης του τελικού προϊόντος. Η σχεδίαση καθορίζει τη συνολική αρχιτεκτονική της εφαρμογής, τον διαχωρισμό των υποσυστημάτων, τις μεταξύ τους σχέσεις και τον τρόπο με τον οποίο αυτά επικοινωνούν. Στην παρούσα εφαρμογή επιλέχθηκε η αρχιτεκτονική client-server, η οποία παρέχει σαφή διαχωρισμό λειτουργιών και επεκτασιμότητα.

Η αρχιτεκτονική client server διαχωρίζει την εφαρμογή σε δύο κύρια μέρη:

- Frontend: Είναι το τμήμα της εφαρμογής που εκτελείται στο πρόγραμμα περιήγησης του χρήστη. Είναι υπεύθυνο για την παρουσίαση της πληροφορίας και τη συλλογή εισόδου από τον χρήστη.
- Backend: Είναι υπεύθυνο για την επεξεργασία των αιτημάτων, την επικοινωνία με τη βάση δεδομένων και την αποστολή των απαραίτητων αποκρίσεων στον πελάτη.

Η ροή των δεδομένων πραγματοποιείται μέσα από HTTP αιτήματα (requests) και απαντήσεις (responses) μεταξύ του frontend και του backend. Η διαδικασία έχει ως εξής:

1. Ο χρήστης υποβάλλει ένα αίτημα μέσω αλληλεπίδρασης με το περιβάλλον της εφαρμογής. Για παράδειγμα σχεδιάζει ένα αυτόματο και επιλέγει την ενέργεια αποθήκευσης ή φόρτωσης.
2. Το frontend, αποστέλλει ένα HTTP αίτημα (GET ή POST) προς τον server. Τα δεδομένα του αυτομάτου μεταφέρονται σε μορφή JSON.
3. Ο server λαμβάνει το αίτημα, το αναλύει και εφόσον χρειάζεται πρόσβαση σε αποθηκευμένα δεδομένα, συνδέεται με τη βάση δεδομένων (PostgreSQL).
4. Κατά την διαδικασία σύνδεσης, το backend πραγματοποιεί έλεγχο στη βάση αναζητώντας το email που δόθηκε, καθώς και αν ο κωδικός πρόσβασης ταιριάζει με το αποθηκευμένο hash.
5. Το backend επιστρέφει στο frontend την αντίστοιχη απάντηση (επιτυχία/αποτυχία, errors κ.λπ.)
6. Το frontend αναλύει την απόκριση και ενημερώνει τη διεπαφή του χρήστη με βάση το αποτέλεσμα (π.χ. επιτυχής σύνδεση, εμφάνιση αποθηκευμένων αυτομάτων, κ.ά.)

Γενικότερα, το frontend ασχολείται με τη διαχείριση καταστάσεων και μεταβάσεων καθώς και την ολοκληρωτική σχεδίαση και αναπαράσταση των αυτομάτων μέσω SVG. Επίσης, η εμφάνιση των αναδυόμενων παραθύρων για αποθήκευση, φόρτωση, σύνδεση, εγγραφή αλλά και υποστήριξη θεμάτων (φωτεινό/σκοτεινό), γλωσσών και διαδραστικών στοιχείων είναι συστατικά του frontend.

Το backend αναλαμβάνει την υλοποίηση των endpoints για σύνδεση (/login), εγγραφή (/register), αποθήκευση (/save) και φόρτωση αυτομάτων (/load) αλλά και τον έλεγχο συμβολοσειρών πάνω σε ένα αυτόματο (/simulate). Τέλος, πραγματοποιεί την επικοινωνία με τη βάση, για την αποθήκευση στον πίνακα χρηστών ή στον πίνακα των αυτομάτων.

4.3 Frontend

4.3.1 HTML

Η βασική δομή της εφαρμογής είναι γραμμένη σε HTML. Χρησιμοποιήθηκε για τη δημιουργία της δομής των στοιχείων της εφαρμογής, όπως τα κουμπιά, οι φόρμες και τα πεδία εισαγωγής. Το έγγραφο ακολουθεί τη δομή HTML5 και περιλαμβάνει τα εξής βασικά τμήματα:

```
<!DOCTYPE html>
<html lang="en">

> <head>···
</head>

> <body>···
</body>

</html>
```

Το αρχείο περιλαμβάνει αρχικά το τμήμα <head> , όπου ορίζονται η κωδικοποίηση χαρακτήρων (UTF-8), οι ρυθμίσεις responsive σχεδίασης (viewport), ο τίτλος της σελίδας, το favicon (εικονίδιο της καρτέλας), καθώς και η σύνδεση του εγγράφου HTML με δύο αρχεία CSS για μορφοποίηση του.

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="author" content="Evangelos Georgosoulis">
    <title>Finite Automata Designer</title>
    <link rel="icon" type="image/png" href="images/process.png">
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="modalSaveLoad.css">
</head>
```

Το κυρίως περιεχόμενο της εφαρμογής, περιλαμβάνεται στην ετικέτα <body> και χωρίζεται σε επιμέρους containers όπως header, main container, footer.

Το τμήμα header αναφέρεται στο κεντρικό μενού που περιέχει το κουμπί για σύνδεση/εγγραφή χρήστη, τον τίτλο της εφαρμογής, τα radio buttons για επιλογή

τύπου αυτομάτου (DFA, NFA) και τα κουμπιά για την εναλλαγή γλώσσας και θέματος.

Το main container περιέχει το αριστερό μενού (καταστάσεις, μεταβάσεις), το δεξί μενού (δοκιμές συμβολοσειρών) και τον κεντρικό καμβά (SVG container).

Το μενού στα αριστερά είναι υπεύθυνο για τη διαχείριση καταστάσεων και μεταβάσεων. Υπάρχουν οι επιλογές για προσθήκη/αφαίρεση καταστάσεων, ορισμό αρχικής/τελικής κατάστασης, ο χρωματισμός και για τις μεταβάσεις αντίστοιχα, η προσθήκη/αφαίρεση, η προσθήκη επαναληπτικού βρόχου (self-loop transition) και η επεξεργασία συμβόλου εισόδου μιας μετάβασης. Επίσης υπάρχουν οι επιλογές για αποθήκευση, φόρτωση καθώς και για καθαρισμό του καμβά SVG, οι οποίες και εμπλουτίζονται με αναδυόμενες βοηθητικές ετικέτες (tooltips) μέσω της χρήσης της ιδιότητας title, διευκολύνοντας έτσι την κατανόηση της λειτουργίας κάθε κουμπιού.

```
<div class="actions-menu">
    <h2 id="states-heading">States</h2>
    <button id="addState" title="Add a new state to the automaton">Add</button>
    <button id="removeState" title="Delete the last state of the automaton">Remove</button>
    <button id="setInitialState" title="Mark selected state as initial">Set Initial</button>
    <button id="setFinalState" title="Mark selected state as final">Set Final(s)</button>
    <button id="colorState" title="Change state color">Set Color</button>

    <h2 id="transitions-heading">Transitions</h2>
    <button id="addTransition" title="Add transition between states">Add</button>
    <button id="removeTransition" title="Delete selected transition">Remove</button>
    <button id="selfLoopTransition" title="Add self-loop transition to a state">Self Loop</button>
    <button id="editTransition" title="Edit transition symbol">Edit</button>

    <h2 id="actions-heading">Actions</h2>
    <button id="saveFA" title="Save automaton to a file">Save</button>
    <button id="loadFA" title="Load automaton from a file">Load</button>
    <input type="file" id="fileInput" accept=".json" style="display: none;">
    <button id="clearSvgArea" title="Clear the workspace">Clear</button>
</div>
```

Το μενού για τη δοκιμή συμβολοσειρών περιλαμβάνει εκτός από τα μηνύματα κειμένου, το πεδίο εισαγωγής συμβολοσειρών και το κουμπί “Δοκιμή” για την προσομοίωση του αυτομάτου.

```
<div class="test-menu">
    <h2 id="testAutomaton">Test Automaton</h2>
    <p id="acceptedStrings">Type strings for testing</p>
    <input type="text" id="testStrings" placeholder="e.g. 010, 111, 1001">
    <button id="testFA" title="Test strings for the current automaton">Test</button>
    <p id="testResults">Test Results:</p>
</div>
```

Τέλος, ο πίνακας SVG, που πραγματοποιείται η γραφική σχεδίαση με χρήση SVG σχημάτων για την αναπαράσταση του αυτομάτου.

```
<div id="svg-container">
    <svg id="svg-area"></svg>
</div>
```

Επιπροσθέτως, το τμήμα <body> περιέχει και δηλώσεις για τα αναδυόμενα παράθυρα (modal) που εμφανίζονται κατά τη χρήση της εφαρμογής. Κάθε modal περιέχει το κουμπί “X” για κλείσιμο, τις κατάλληλες φόρμες και checkboxes για τις λειτουργίες αποθήκευσης, φόρτωσης και σύνδεσης/εγγραφής του χρήστη.

Τέλος, υπάρχει το τμήμα <footer>, που περιλαμβάνει κάποιες επιπλέον πληροφορίες και την αναφορά στην πηγή για τα εικονίδια, καθώς και οι συνδέσεις του εγγράφου HTML, με τα αρχεία JavaScript που θα αναλάβουν τη λογική πίσω από την εφαρμογή.

```
<footer>
    <p>
        © 2025 Evangelos Georgosoulis
        -
        <a href="https://www.flaticon.com/" target="_blank" rel="noopener noreferrer" style="color: inherit;">Icons
            by Flaticon</a>
    </p>
</footer>
<script src="main.js"></script>
<script src="saveLoad.js"></script>
```

4.3.2 CSS

Για την εμφάνιση και τη μορφοποίηση των στοιχείων HTML χρησιμοποιήθηκαν αρχεία CSS, τα οποία αναφέρονται σε διαφορετικές οντότητες. Αρχικά, στο κύριο αρχείο δηλώνονται μεταβλητές μέσω :root με τις οποίες καθορίζονται χρώματα, σκιές και περιγράμματα για την εμφάνιση της εφαρμογής. Αντίστοιχες μεταβλητές υπάρχουν και για τη σκοτεινή λειτουργία (body.dark-mode), οι οποίες εναλλάσσονται με την επιλογή αλλαγής θέματος της εφαρμογής.

Τα κύρια τμήματα που αναπτύσσονται στο αρχείο είναι τα body, header και SVG area. Συγκεκριμένα, στο τμήμα body περιλαμβάνεται το χρώμα του φόντου, η γραμματοσειρά αλλά και ο ορισμός ως flex container για να υποστηρίζει κάθετη διάταξη με τα υπόλοιπα τμήματα (header, container, footer).

```
body {
    font-family: 'Inter', 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: var(--bg-color);
    color: var(--text-color);
    box-sizing: border-box;
    display: flex;
    flex-direction: column;
    min-height: 100vh;
}
```

Το header περιλαμβάνει επίσης χρώμα φόντου, κεντρική στοίχιση κειμένου αλλά και ρυθμίσεις για τη δομή και την τοποθέτηση ολόκληρου του τμήματος.

```css
header {
    background-color: var(--header-color);
    border-bottom: 1px solid var(--border-color);
    padding: 1rem 2rem;
    display: flex;
    flex-direction: column;
    align-items: center;
    gap: 0.5rem;
    text-align: center;
    box-shadow: 0 2px 4px var(--shadow-light);
}
```

Το svg-area ορίζει την σωστή εμφάνιση του πίνακα για τη σχεδίαση των αυτομάτων και προσθέτει διακεκομμένη γραμμή στα όρια, πλέγμα στο εσωτερικό του αλλά και οριζόντια μπάρα κύλισης (scrollbar) για την επέκταση του χώρου σε περίπτωση αυτομάτων με πολλές καταστάσεις.

```css
#svg-area {
    flex-grow: 1;
    height: auto;
    min-height: 100%;
    margin: 0 0rem;
    position: relative;
    background-color: var(--svg-bg);
    box-shadow: inset 0 0 10px rgba(255, 255, 255, 0.05);
    /* grid */
    background-image: linear-gradient(var(--grid-color, rgba(0, 0, 0, 0.1)) 1px, transparent 1px),
        linear-gradient(90deg, var(--grid-color, rgba(0, 0, 0, 0.1)) 1px, transparent 1px);
    background-size: 20px 20px;
    min-width: 3000px;
    display: block;
}

#svg-container {
    overflow-x: auto;
    overflow-y: hidden;
    width: 100%;
    flex-grow: 1;
    border: 2px dashed var(--svg-border);
    border-radius: 12px;
    background-color: var(--svg-bg);
    position: relative;
}
```

Τέλος, υπάρχουν και μορφοποιήσεις συγκεκριμένων στοιχείων όπως τα κουμπιά, που διαθέτουν σκιές, στρογγυλεμένες γωνίες και μεταβάσεις κατά το hover, τα inputs και radio buttons, τα toggle buttons και τα κείμενα στις παραγράφους.

Το δεύτερο αρχείο CSS, ενσωματώνει τη μορφοποίηση των στοιχείων διεπαφής χρήστη (UI) με έμφαση στην ευχρηστία και την προσβασιμότητα. Στην αρχή, ορίζονται οι μεταβλητές χρωμάτων για φωτεινό και σκοτεινό θέμα. Επιπλέον, το αρχείο καθορίζει τη μορφοποίηση και συμπεριφορά διαλόγων τύπου modal για τη διαχείριση εισόδου χρήστη (login/register), καθώς και για τη φόρτωση και διαχείριση αυτομάτων από τον εξυπηρετητή. Οι διάλογοι υλοποιούνται με την εντολή `position:fixed` ώστε να καλύπτουν μέρος της οθόνης και να εστιάζουν στην προσοχή του χρήστη, ενώ η χρήση animation (fadeIn) συμβάλει στην ομαλή τους μετάβαση.

```css
.modal-content {
    animation: fadeIn 0.3s ease-in-out;
}

@keyframes fadeIn {
    from {
        opacity: 0;
        transform: translateY(-20px);
    }

    to {
        opacity: 1;
        transform: translateY(0);
    }
}
```

Τα επιμέρους στοιχεία των διαλόγων, όπως τα πεδία εισόδου, οι καρτέλες επιλογής και τα κουμπιά, ακολουθούν πρότυπα σχεδίασης που στοχεύουν στη διευκόλυνση της κατανόησης και της λειτουργικής συνέπειας της διεπαφής. Τέλος, η χρήση προσαρμοσμένων scrollbar για τις λίστες εξυπηρετεί αισθητικούς αλλά και λειτουργικούς σκοπούς, ειδικά όταν διαχειρίζονται μεγάλες λίστες δεδομένων (αυτόματα αποθηκευμένα στον server).

4.3.3 JavaScript

Η JavaScript αποτελεί το λειτουργικό υπόβαθρο της διεπαφής χρήστη στην εφαρμογή, καθώς συντονίζει τις ενέργειες σχεδίασης, αλληλεπίδρασης, αποθήκευσης και επικοινωνίας με τον server. Στο frontend αξιοποιούνται δύο βασικά αρχεία: το main.js, το οποίο συγκεντρώνει τη λογική σχεδίασης και χειρισμού αυτομάτων και το saveLoad.js, που αναλαμβάνει τη λειτουργία αποθήκευσης και φόρτωσης.

Το main.js αρχείο αποτελεί τον βασικό πυρήνα της εφαρμογής. Ξεκινά με την υλοποίηση των event listeners που σχετίζονται με τη δημιουργία καταστάσεων (states). Ο χρήστης μπορεί να προσθέσει μια νέα κατάσταση με το κουμπί «Προσθήκη», το οποίο δημιουργεί δυναμικά SVG στοιχεία (<circle> & <text>) μέσα στον καμβά (svg-area). Χρησιμοποιείται η συνάρτηση getNextStateId() για τον υπολογισμό του ονόματος κάθε κατάστασης (q0, q1, …) και στη συνέχεια σχεδιάζεται η κατάσταση όπως εμφανίζεται και παρακάτω:

```
//dhmiourgia katastashs
const circle = document.createElementNS("http://www.w3.org/2000/svg", "circle");
circle.setAttribute("cx", posX);
circle.setAttribute("cy", posY);
circle.setAttribute("r", "30");
circle.setAttribute("stroke", "black");
circle.setAttribute("stroke-width", "2");
circle.setAttribute("fill", "orange");

circle.setAttribute("data-id", stateId);
```

Κάθε φορά, δημιουργούνται τα απαραίτητα attributes για την οπτική αναπαράσταση της κατάστασης, σε ανανεωμένη τοποθεσία στον καμβά. Το όνομα της κατάστασης αποδίδεται μέσω SVG <text> στοιχείου, το οποίο περιέχει το μοναδικό αναγνωριστικό.

```
//onoma ths katastashs
const text = document.createElementNS("http://www.w3.org/2000/svg", "text");
text.setAttribute("x", posX);
text.setAttribute("y", posY);
text.setAttribute("font-size", "18");
text.setAttribute("fill", "black");
text.setAttribute("text-anchor", "middle");
text.setAttribute("dominant-baseline", "middle");
text.textContent = stateId;
```

Τα <circle> και <text> ενσωματώνονται σε ένα κοινό SVG <g> (group) στοιχείο, το οποίο επιτρέπει τη συνδυασμένη διαχείριση πάνω στα γραφικά στοιχεία (μετακίνηση, επιλογή, διαγραφή) ως μία εννιαίας οντότητας.

Η διαχείριση μετακινήσεων υποστηρίζεται μέσω DOM events (mousedown, mousemove, mouseup, click). Ο χρήστης μπορεί να επιλέξει μια κατάσταση, να την ορίσει ως αρχική ή ως τελική και να αλλάξει το χρώμα της. Στη συνέχεια υλοποιείται και η λειτουργικότητα των μεταβάσεων. Οι μεταβάσεις σχεδιάζονται ως γραμμές μεταξύ των μεταβάσεων, με ετικέτες συμβόλων. Η εφαρμογή υποστηρίζει και τον ορισμό πολλαπλών συμβόλων εισόδου σε μετάβαση που καταλήγει στην ίδια κατάσταση (χωρισμένα με κόμμα), πχ. a, b, c.

Σημαντικός είναι και ο έλεγχος εγκυρότητας που εφαρμόζεται κατά τη δημιουργία μεταβάσεων. Αν ο χρήστης έχει επιλέξει τύπο αυτομάτου ΝΠΑ, τότε δεν επιτρέπεται η εισαγωγή κενού συμβόλου σε μετάβαση ή ο ορισμός πολλαπλών μεταβάσεων, από μία κατάσταση, για το ίδιο σύμβολο. Σε κάθε περίπτωση εμφανίζεται κατάλληλο μήνυμα (alert) που ενημερώνει τον χρήστη.

```
//me epilogh dfa den epitrepontai kenes metavaseis kai metavaseis me idio sumvolo
if (automatonType === "DFA") {
    if (!transitionLabel) {
        alert(getTranslation("alertEmptyTransitions"));
        return;
    }

    const fromId = selectedStates[0].getAttribute("data-id");
    const duplicate = document.querySelector(`.transition[data-from="${fromId}"][data-symbol="${transitionLabel}"]`);
    if (duplicate) {
        alert(getTranslation("alertSameSymbolTransition"))
        return;
    }
}
```

Πέρα από τη σχεδίαση του αυτομάτου, το αρχείο περιλαμβάνει και τη λειτουργία ελέγχου αποδοχής συμβολοσειράς, σύμφωνα με τη λογική των πεπερασμένων

αυτομάτων. Ο χρήστης εισάγει πλήθος συμβολοσειρών στο κατάλληλο πεδίο και με την επιλογή «Δοκιμή», ενεργοποιείται η αποστολή του αυτομάτου στον διακομιστή για προσομοίωση.

Για τη διαδικασία αυτή, χρησιμοποιείται η συνάρτηση getAutomatonData(), η οποία συλλέγει όλα τα δεδομένα του σχεδιασμένου αυτομάτου από τον SVG καμβά. Πιο συγκεκριμένα, διαβάζει τις καταστάσεις, τις μεταβάσεις, τις ιδιότητες (αρχική/τελική κατάσταση), τον τύπο (DFA, NFA), καθώς και τη δομή του γραφήματος. Το αποτέλεσμα είναι ένα πλήρες αντικείμενο JSON που αποστέλλεται μέσω fetch() στον server, ο οποίος θα εφαρμόσει τον αντίστοιχο αλγόριθμο προσομοίωσης και θα επιστρέψει τη λίστα με τις συμβολοσειρές και το αποτέλεσμα αποδοχής ή απόρριψης της κάθε συμβολοσειράς.

Το αρχείο διαχειρίζεται επίσης τη διεπαφή χρήστη για θέματα εμφάνισης, όπως την αλλαγή γλώσσας, την εναλλαγή σκοτεινού/ φωτεινού θέματος και την αλληλεπίδραση με modals για αποθήκευση, φόρτωση και σύνδεση χρήστη.

Η δυναμική ενημέρωση της γλώσσας εμφάνισης πραγματοποιείται με την συνάρτηση getTranslation() η οποία σε συνδυασμό με το αρχείο translations.json, που περιέχει τις μεταφράσεις σε αγγλικά και ελληνικά, αλλάζει τα κείμενα που εμφανίζονται ανάλογα την επιλογή του χρήστη. Με την επιλογή του κουμπιού theme-toggle, υπεύθυνου για το σκοτεινό/φωτεινό θέμα, αλλάζουν τα χρώματα της εφαρμογής μέσω CSS, καθώς και η εμφάνιση του εικονιδίου πάνω στο κουμπί. Τέλος, το αρχείο χειρίζεται λειτουργίες εμφάνισης και απόκρυψης των παραθύρων modal, ανάλογα με τις κινήσεις του χρήστη.

Κατά τη σχεδίαση του αυτομάτου, η εφαρμογή χρησιμοποιεί SVG (Scalable Vector Graphics) ως τεχνολογία απεικόνισης των στοιχείων ενός αυτομάτου στον καμβά. Το SVG αποτελεί γλώσσα σήμανσης για γραφικά βασισμένη σε XML και υποστηρίζεται πλήρως από όλους τους σύγχρονους browsers. Επιτρέπει τη δημιουργία και τον χειρισμό γραφικών αντικειμένων όπως κύκλοι, γραμμές, καμπύλες, κείμενα, καθώς και τη σύνδεση τους με event listeners, γεγονός που καθιστά τη συγκεκριμένη τεχνολογία ιδανική για διαδραστικές εφαρμογές όπως αυτή. Στην παρούσα εφαρμογή, κάθε κατάσταση απεικονίζεται ως SVG <circle> και συνοδεύεται από ένα <text> για το όνομα της (q0, q1, κ.ο.κ.). Οι μεταβάσεις σχεδιάζονται ως καμπύλες με βέλη και τα σύμβολα εισόδου προβάλλονται με <text>

μέσα σε <rect> ως φόντο, πάνω από τη γραμμή (path). Η χρήση του SVG, σε συνδυασμό με JavaScript, επιτρέπει την πλήρως δυναμική δημιουργία, μετακίνηση και διαγραφή γραφικών στοιχείων, χωρίς την ανάγκη ανανέωσης της σελίδας κάθε φορά. Επιπλέον, εξασφαλίζει καθαρότητα στην εμφάνιση, ευκολία στην αλληλεπίδραση και αποτελεί μία ιδανική λύση ακόμη και σε πολύπλοκα πεπερασμένα αυτόματα, με μεγάλο πλήθος καταστάσεων και μεταβάσεων.

Πέρα από τη λογική της σχεδίασης του αυτομάτου, η εφαρμογή περιλαμβάνει και το αρχείο saveLoad.js, το οποίο διαχειρίζεται τις λειτουργίες αποθήκευσης και φόρτωσης αυτομάτων, είτε τοπικά είτε απομακρυσμένα. Συγκεκριμένα, μόλις πατηθεί η επιλογή «Αποθήκευση» ή «Φόρτωση», αναγνωρίζεται μέσω event listener και εμφανίζεται το αντίστοιχο αναδυόμενο παράθυρο modal.

Για την περίπτωση του παραθύρου που σχετίζεται με την αποθήκευση, πραγματοποιείται έλεγχος ως προς το όνομα που δίνει ο χρήστης στο αρχείο και ως προς το αν υπάρχει σχεδιασμένο αυτόματο στον καμβά για να αποθηκευτεί. Στη συνέχεια, εάν έχει επιλεχθεί η τοπική αποθήκευση, τότε μέσω Blob API κατεβαίνει το αρχείο σε μορφή JSON στον υπολογιστή του χρήστη με το όνομα που έχει πληκτρολογήσει στο προηγούμενο πεδίο. Αντίθετα, αν έχει επιλεχθεί η αποθήκευση στον server πραγματοποιείται ένα HTTP POST αίτημα στον διακομιστή, στο route /user/save, και στέλνεται το αυτόματο ως JSON προς αποθήκευση στη βάση. Ανάλογα με την απάντηση του διακομιστή, εμφανίζεται αντίστοιχο alert που ενημερώνει κατάλληλα τον χρήστη για την επιτυχία ή αποτυχία της ενέργειας.

Όσον αφορά την επιλογή της φόρτωσης αυτομάτου, εάν ο χρήστης διαλέξει την τοπική φόρτωση θα ανοίξει ο file explorer του λειτουργικού συστήματος (μέσω FileReader) για να επιλέξει το αρχείο. Γίνονται αποδεκτά μόνο αρχεία τύπου JSON, το οποία και δίνονται ως όρισμα στην συνάρτηση loadAutomaton() και ξεκινά η εκ νέου σχεδίαση του αυτομάτου στον καμβά με βάση τα αποθηκευμένα στοιχεία.

Στην περίπτωση της φόρτωσης στον server, στέλνεται ένα αίτημα GET, μέσω fetch() στο endpoint /user/automata, με το email του χρήστη και ελέγχεται αν υπάρχουν αποθηκευμένα αυτόματα. Αν υπάρχουν, επιστρέφεται η λίστα με τα αποθηκευμένα αυτόματα του χρήστη, τα οποία προβάλλονται δυναμικά σε ξεχωριστό modal, το οποίο περιέχει και επιλογές για φόρτωση σε κάθε αυτόματο.

Το αρχείο διαχειρίζεται επιπλέον λειτουργίες, όπως η εμφάνιση της λίστας των αποθηκευμένων αυτομάτων χρήστη με δυνατότητα διαγραφής και μετονομασίας των αυτομάτων που βρίσκονται στη βάση, καθώς και ελέγχους όπως η αποτροπή ταυτόχρονης επιλογής τοπικής και απομακρυσμένης φόρτωσης. Τέλος, πραγματοποιείται έλεγχος για το αν ο χρήστης είναι συνδεδεμένος, διαφορετικά οι επιλογές που σχετίζονται με τον διακομιστή παραμένουν απενεργοποιημένες.

4.4 Backend

Το backend μέρος της εφαρμογής έχει υλοποιηθεί σε JavaScript μέσω του Node.js, με χρήση του framework Express για την ανάπτυξη RESTful υπηρεσιών και τη διαχείριση των αιτημάτων από το frontend. Η λογική του διακομιστή αναλύεται στα αρχεία server.js, auth.js, saveRoutes.js, database.js, καθώς και τα αρχεία για τον έλεγχο των συμβολοσειρών dfaSimulator.js και nfaSimulator.js.

4.4.1 Node και Express

Η τεχνολογία Node.js σε συνδυασμό με το framework Express.js χρησιμοποιήθηκαν για την υλοποίηση του backend κομματιού της εφαρμογής, επιτρέποντας την επεξεργασία αιτημάτων, τη σύνδεση με τη βάση δεδομένων και την εκτέλεση της λογικής των ΠΑ.

Το βασικό αρχείο του backend είναι το server.js, το οποίο λειτουργεί ως σημείο εισόδου του διακομιστή. Μέσα σε αυτό αρχικοποιείται το Express app, εισάγονται όλες οι απαραίτητες εξωτερικές βιβλιοθήκες (express, cors, dotenv κ.ά.) και δηλώνονται τα middleware που επεξεργάζονται τα εισερχόμενα δεδομένα σε μορφή JSON. Επιπλέον, φορτώνονται και συνδέονται τα επιμέρους αρχεία που σχετίζονται με την αυθεντικοποίηση (auth.js), τις λειτουργίες αποθήκευσης (saveRoutes.js) και τον έλεγχο συμβολοσειρών των αυτομάτων (dfaSimulator & nfaSimulator). Τέλος, το αρχείο αυτό εκκινεί τον διακομιστή στην καθορισμένη θύρα 3000 μέσω της μεθόδου app.listen() και καθορίζει τις βασικές REST διαδρομές της εφαρμογής.

Ιδιαίτερη σημασία έχει η ενσωμάτωση των αρχείων dfaSimulator.js & nfaSimulator.js, τα οποία περιλαμβάνουν τους αλγορίθμους για τον έλεγχο

αποδοχής συμβολοσειρών με βάση τα μαθηματικά μοντέλα των ντετερμινιστικών και μη-ντετερμινιστικών πεπερασμένων αυτομάτων.

Το αρχείο dfaSimulator.js υλοποιεί τη βασική συνάρτηση simulateDFA(), η οποία δέχεται ως παραμέτρους το αντικείμενο που περιγράφει το DFA (με καταστάσεις, μεταβάσεις, αρχική και τελικές καταστάσεις) και μία συμβολοσειρά από αυτές που έχει εισάγει ο χρήστης στο πεδίο για έλεγχο. Η συνάρτηση εντοπίζει την αρχική κατάσταση, διατρέχει τη συμβολοσειρά χαρακτήρα προς χαρακτήρα και εφαρμόζει τη συνάρτηση μετάβασης με βάση το τρέχον σύμβολο. Αν σε κάποιο σημείο δεν υπάρχει διαθέσιμη μετάβαση για το συγκεκριμένο σύμβολο, η συμβολοσειρά απορρίπτεται. Αν μετά την πλήρη διαδικασία, η τελική κατάσταση που βρίσκεται η συνάρτηση ανήκει στο σύνολο των καταστάσεων αποδοχής, τότε η συμβολοσειρά γίνεται αποδεκτή. Ο χρήστης έχει τη δυνατότητα να εισάγει πολλαπλές συμβολοσειρές για έλεγχο, χωρισμένες με κόμμα και γι' αυτό και η συνάρτηση τρέχει για κάθε συμβολοσειρά που εισάγεται μεμονωμένα.

Αντίστοιχα, το αρχείο nfaSimulator.js, υλοποιεί τη συνάρτηση simulateNFA(), με ίδιες παραμέτρους εισόδου με αυτές της simulateDFA(). Η συνάρτηση επιτρέπει πολλαπλές διαδρομές αλλά και την ειδική μεταχείριση των έψιλον-μεταβάσεων (ε) . Η λογική του NFA βασίζεται στην έννοια της παράλληλης εξέτασης όλων των δυνατών καταστάσεων. Κατά την εκτέλεση, η προσομοίωση δεν ξεκινά μόνο από την τυπικά ορισμένη αρχική κατάσταση, αλλά από το σύνολο των καταστάσεων που είναι προσβάσιμες από αυτή μέσω ε-μεταβάσεων. Σε κάθε βήμα, υπολογίζεται ξανά το σύνολο αυτών των καταστάσεων, έτσι ώστε να συμπεριληφθούν και οι άμεσες μεταβάσεις χωρίς κάποιο σύμβολο εισόδου. Η συμβολοσειρά γίνεται αποδεκτή αν μετά τον έλεγχο όλων των χαρακτήρων της συμβολοσειράς, τουλάχιστον μία από τις ενεργές καταστάσεις είναι τελική. Ο αλγόριθμος είναι πιο σύνθετος από αυτόν για τα DFA, καθώς απαιτεί αναδρομική επεξεργασία πολλαπλών διαδρομών μέσα στο αυτόματο, λόγω της ύπαρξης κενών μεταβάσεων.

Παρότι τα αρχεία αυτά δεν περιλαμβάνουν λειτουργίες Express, ενσωματώνονται στο backend και αξιοποιούνται από τα routes του διακομιστή για την επεξεργασία δεδομένων που αποστέλλονται από τον χρήστη.

Η διαχείριση των αποθηκευμένων αυτομάτων υλοποιείται στο αρχείο saveRoutes.js, το οποίο περιέχει όλες τις διαδρομές (routes) που σχετίζονται με τις

βασικές λειτουργίες αποθήκευσης, φόρτωσης, προβολής και διαγραφής. Κάθε route ακολουθεί τη RESTful λογική, με χρήση μεθόδων HTTP όπως GET, POST, PUT και DELETE. Για παράδειγμα, η διαδρομή POST /save λαμβάνει ένα JSON αντικείμενο που περιλαμβάνει το όνομα του αυτομάτου, τον τύπο του και όλα τα στοιχεία της σχεδίασης (καταστάσεις, μεταβάσεις κ.λπ.). Το αντικείμενο αυτό μετατρέπεται σε string μέσω της μεθόδου JSON.stringify() και αποθηκεύεται στη βάση στη στήλη json_data μέσω SQL queries. Στο ίδιο αρχείο, η διαδρομή GET /load επιστρέφει όλα τα αυτόματα που ανήκουν στον αυθεντικοποιημένο χρήστη, ενώ με τη DELETE /delete/:id επιτρέπεται η διαγραφή ενός συγκεκριμένου αυτομάτου με βάση το μοναδικό του αναγνωριστικό.

```
//save to automato sth vash
router.post("/save", async (req, res) => {···
});

//anakthsh twn saved automatwn tou sundedemenou xrhsth
router.post("/automata", async (req, res) => {···
});

//fortwsh automatou
router.get("/loadone", async (req, res) => {···
});

//rename to onoma tou arxeiou sth vash
router.put("/edit", async (req, res) => {···
});

//diagrafh arxeiou sth vash
router.delete("/delete", async (req, res) => {···
});

//epistrefei th lista me ta automata sumfwna me id
router.get("/load", async (req, res) => {···
});
```

4.4.2 PostgreSQL

Η εφαρμογή χρησιμοποιεί τη σχεσιακή βάση δεδομένων PostgreSQL για την αποθήκευση δεδομένων χρηστών και αυτομάτων. Η επιλογή της PostgreSQL έγινε λόγω της αξιοπιστίας, της υποστήριξης πολύπλοκων ερωτημάτων, της επεκτασιμότητας και της δωρεάν διάθεσης της ως open-source λογισμικό. Επιπλέον, η υποστήριξη τύπου JSON την καθιστά ιδανική για εφαρμογές που απαιτούν αποθήκευση δεδομένων όπως στην παρούσα περίπτωση αναπαριστούν ένα αυτόματο σε πλήρη μορφή.

Η βάση δεδομένων περιλαμβάνει δύο πίνακες.

1. Τον πίνακα users που περιέχει την ηλεκτρονική διεύθυνση και τον κωδικό πρόσβασης του χρήστη σε hashed μορφή.

	id [PK] integer	email character varying (255)	password_hash text
1	1	test@mail.com	$2b$10$P6Qrqd3I9Mvd4VsH3PUBIOJnyNb9zm6svKjQj89NZVnrWrZn7i60e
2	2	user@gmail.com	$2b$10$sRHwMnSeWHXv1Zu3eGwp2.qb1Whawjg72.tth1gmpJAP5KDNF...
3	3	user2@example.com	$2b$10$gJwaH22cHEeyReL1JswcauEoQXW8bCkf/EtF2mLwEiR9CxOAiG...

2. Τον πίνακα automata που περιέχει το όνομα του αυτομάτου, τον τύπο του, καθώς και όλα τα δεδομένα σχεδίασης και μεταβάσεων σε μορφή JSON.

id [PK] integer	user_id integer	name character var	type character var	created_at timestamp wi	json_data jsonb
7	7	test	NFA	2025-04-...	{"type": "NFA", "states": [{"x": 55, "y": 38, "id": "q0", "color": "orange", "isFinal": false, "isInitial": true}, {"x": 213, "y": 83, "id": "q1", "color"
11	5	example1	DFA	2025-04-...	{"type": "DFA", "states": [{"x": 50, "y": 100, "id": "q0", "color": "orange", "isFinal": false, "isInitial": false}, {"x": 130, "y": 100, "id": "q1", "co
13	10	(0|1)*	DFA	2025-04-...	{"type": "DFA", "states": [{"x": 268, "y": 190, "id": "q0", "color": "lightgreen", "isFinal": false, "isInitial": true}, {"x": 511, "y": 189, "id": "q1",

Η επικοινωνία της εφαρμογής Node.js με την PostgreSQL επιτυγχάνεται μέσω του πακέτου pg, το οποίο παρέχει client API για αποστολή εντολών SQL. Το αρχείο database.js ορίζει ένα connection pool, δηλαδή ένα σύνολο από επαναχρησιμοποιούμενες συνδέσεις, ώστε να βελτιστοποιείται η απόδοση του διακομιστή και να αποφεύγεται η δημιουργία νέας σύνδεσης σε κάθε αίτημα. Η σύνδεση διαμορφώνεται δυναμικά με βάση τις μεταβλητές περιβάλλοντος από το αρχείο .env όπως DB_USER, DB_PASSWORD, DB_NAME, κ.λπ.

```
require("dotenv").config();
const { Pool } = require("pg");

const pool = new Pool({
    user: process.env.DB_USER,
    host: process.env.DB_HOST,
    database: process.env.DB_DATABASE,
    password: process.env.DB_PASSWORD,
    port: process.env.DB_PORT
});
module.exports = pool;
```

Μέσω του pool μπορούν να εκτελούνται ασύγχρονα SQL queries με τη χρήση await pool.query(). Κατά την αποθήκευση ενός νέου αυτομάτου από τον χρήστη, η εφαρμογή αποστέλλει ένα POST αίτημα στο route /save το οποίο οδηγεί στην εκτέλεση SQL ερωτήματος τύπου INSERT. Το query καταχωρεί το user_id του χρήστη, το name, το type και το πλήρες json_data του αυτομάτου:

```
await pool.query(
    "INSERT INTO automata (user_id, name, type, json_data) VALUES ($1, $2, $3, $4)",
    [userId, name, automaton.type, JSON.stringify(automaton)]
);
```

Αντίστοιχα, κατά τη φόρτωση αποθηκευμένων αυτομάτων ενός χρήστη, αποστέλλεται ένα αίτημα GET /load, το οποίο οδηγεί στην εκτέλεση εντολής ανάκτησης με βάση το user_id:

```
const result = await pool.query("SELECT id, name FROM automata WHERE user_id = $1", [userId]);
```

Η χρήση της σύνταξης $1, $2, κ.λπ. στο SQL επιτρέπει την παραμετροποίηση των ερωτημάτων, προσφέροντας αυξημένη ασφάλεια (κατά των SQL injection επιθέσεων) και καθαρότερο διαχωρισμό λογικής και δεδομένων. Οι παράμετροι αντικαθίστανται με ασφάλεια από τις τιμές που δίνονται σε πίνακα στο pool.query() και δεν ενσωματώνονται απευθείας στο query string.

4.4.3 Αυθεντικοποίηση Χρήστη

Η διαδικασία αυθεντικοποίησης χρηστών (user authentication) αποτελεί κρίσιμο τμήμα της εφαρμογής, καθώς διασφαλίζει την εξατομικευμένη πρόσβαση των χρηστών και επιτρέπει την προστατευμένη αποθήκευση και ανάκτηση των προσωπικών τους δεδομένων, όπως τα σχεδιασμένα αυτόματα. Παράλληλα, υποστηρίζει τη διάκριση μεταξύ εγγεγραμμένου χρήστη και ανώνυμου χρήστη, παρέχοντας διαφορετικά επίπεδα λειτουργικότητας.

Η υλοποίηση της αυθεντικοποίησης πραγματοποιείται στο backend της εφαρμογής, με χρήση της τεχνολογίας Node.js και της βιβλιοθήκης bcrypt. Η bcrypt επιτρέπει την ασφαλή διαχείριση των κωδικών πρόσβασης, εφαρμόζοντας αλγόριθμο κατακερματισμού (hashing) με χρήση διαδικασίας salting και πολλαπλών γύρων επεξεργασίας. Έτσι, ακόμη και σε περιπτώσεις διαρροής της βάσης δεδομένων, οι κωδικοί παραμένουν μη αναγνώσιμοι, καθώς δεν αποθηκεύονται ποτέ σε απλή μορφή (plaintext).

Η βασική λογική της αυθεντικοποίησης έχει ενσωματωθεί στο αρχείο auth.js, το οποίο διαχειρίζεται δύο κύρια RESTful endpoints: POST /register για τη δημιουργία νέου χρήστη και POST /login για την σύνδεση υπάρχοντος χρήστη. Κατά την εγγραφή, ο χρήστης εισάγει τη διεύθυνση email και τον κωδικό πρόσβασης της αρέσκειας του. Ο server, πριν αποθηκεύσει τα στοιχεία, ελέγχει εάν το email είναι μοναδικό και έπειτα προχωρά στον κατακερματισμό του κωδικού με τη χρήση της εντολής:

```
const hashedPassword = await bcrypt.hash(password, 10);
```

Η τιμή 10 καθορίζει τον αριθμό των επαναλήψεων του αλγορίθμου, προσφέροντας προστασία απέναντι σε brute-force επιθέσεις. Το παραγόμενο hash αποθηκεύεται στον πίνακα users της PostgreSQL μαζί με το email, μέσω SQL query.

```
await pool.query("INSERT INTO users (email, password_hash) VALUES ($1,$2)",
    [email, hashedPassword]
);
```

Κατά τη σύνδεση, ο χρήστης εισάγει εκ νέου τα στοιχεία του. Ο διακομιστής αναζητά τη διεύθυνση email στη βάση, ανακτά τον σχετικό κατακερματισμένο

κωδικό και συγκρίνει τη νέα είσοδο με τον αποθηκευμένο hash, μέσω της εντολής:

```
const isMatch = await bcrypt.compare(password, user.password_hash)
```

Η σύγκριση γίνεται ασύγχρονα και δεν επιτρέπει αποκρυπτογράφηση του hash. Αν η σύγκριση είναι επιτυχής, επιστρέφεται το user_id και ενεργοποιούνται οι κλειδωμένες προηγουμένως λειτουργίες χρήστη, όπως η αποθήκευση και η φόρτωση αυτομάτων στον server. Εάν αποτύχει, ο server απορρίπτει το αίτημα και εμφανίζει κατάλληλο μήνυμα.

Το auth.js χειρίζεται εσωτερικά τον έλεγχο των στοιχείων χωρίς τη χρήση tokens ή sessions, χρησιμοποιώντας αντί αυτών έναν απλό μηχανισμό ταυτοποίησης με βάση τον user_id, ο οποίος αποστέλλεται στο frontend και συνοδεύει τα αιτήματα του χρήστη για τις υπόλοιπες λειτουργίες. Στο frontend, η εφαρμογή ελέγχει εάν έχει δοθεί έγκυρο user_id και ενεργοποιεί τις επιλογές που αφορούν τον server.

Σε περίπτωση που ο χρήστης επιλέξει να χρησιμοποιήσει την εφαρμογή χωρίς σύνδεση, ενεργοποιείται η λειτουργία “guest”. Σε αυτήν την περίπτωση είναι διαθέσιμες μόνο οι τοπικές λειτουργίες, όπως η αποθήκευση και η φόρτωση αυτομάτων σε μορφή JSON αρχείου χωρίς αποθήκευση σε cloud και χωρίς πρόσβαση σε λειτουργίες διαχείρισης.

Η αποσύνδεση από τον λογαριασμό εκτελείται τοπικά στο frontend, διαγράφοντας τις τιμές της συνεδρίας και επαναφέροντας την εφαρμογή σε κατάσταση ανώνυμου χρήστη. Απενεργοποιούνται τα κουμπιά που σχετίζονται με τον server και κρύβονται τα στοιχεία του προηγουμένως συνδεδεμένου χρήστη.

5. ΕΠΙΔΕΙΞΗ

5.1 Οδηγός Εγκατάστασης Εφαρμογής

Η εφαρμογή έχει σχεδιαστεί ως διαδικτυακή πλατφόρμα, η οποία μπορεί να λειτουργήσει τοπικά σε περιβάλλον ανάπτυξης (localhost) ή να αναπτυχθεί σε απομακρυσμένο server. Σε αυτό το κεφάλαιο ακολουθεί αναλυτικός οδηγός εγκατάστασης, για χρήση της εφαρμογής σε τοπικό περιβάλλον ανάπτυξης.

5.1.1 Απόκτηση του Πηγαίου Κώδικα

Ο κώδικας της εφαρμογής είναι διαθέσιμος σε δημόσιο αποθετήριο (repository) στο GitHub: https://github.com/egeorgosoulis/finite-automata-web.git.

Υπάρχουν δύο τρόποι να αποκτηθεί ο κώδικας:

1. Μέσω Git:

   Εκτελέστε την παρακάτω εντολή στο τερματικό:

   ```
   git clone https://github.com/egeorgosoulis/finite-automata-web.git
   ```

2. Μέσω αρχείου ZIP:

   Επισκεφθείτε τον σύνδεσμο του αποθετηρίου, επιλέξτε Code και στη συνέχεια Download ZIP. Τέλος, αποσυμπιέστε το αρχείο στον φάκελο της επιλογής σας.

5.1.2 Εγκατάσταση Εξαρτήσεων

Για την ορθή λειτουργία του backend, απαιτείται η εγκατάσταση των απαραίτητων πακέτων (dependencies) του Node.js. Πριν την εκκίνηση, ανοίγουμε το τερματικό στον root φάκελο της εφαρμογής (`cd finite-automata-web`) και εκτελούμε την εντολή :

```
npm install
```

Η εντολή αυτή διαβάζει το αρχείο package.json και εγκαθιστά όλα τα εξαρτώμενα πακέτα (express, bcrypt, pg, κ.ά.) στον φάκελο node_modules.

5.1.3 Ρύθμιση Βάσης Δεδομένων

Η εφαρμογή απαιτεί επίσης τη λειτουργία μιας PostgreSQL βάσης δεδομένων για την αποθήκευση χρηστών και αυτομάτων. Για τη διαχείριση της βάσης χρησιμοποιείται το pgAdmin, ένα γραφικό περιβάλλον που διευκολύνει τη δημιουργία και διαχείριση πινάκων.

Ακολουθούν τα βήματα για τη δημιουργία και εκκίνηση της βάσης σε τοπικό περιβάλλον:

1. Δημιουργήστε μια νέα βάση δεδομένων, με όνομα automata_db.
2. Δημιουργήστε δύο πίνακες:
    a. users: που αποθηκεύει id, email, password_hash
    b. automata: που αποθηκεύει id, user_id, name, type, json_data
3. Ορίστε user_id ως foreign key στον πίνακα users, με ON DELETE CASCADE
4. Ελέγξτε τη σύνδεση και την προσβασιμότητα των πινάκων μέσω του query tool στο pgAdmin.

Η σύνδεση της εφαρμογής με τη βάση διαμορφώνεται στο αρχείο database.js και βασίζεται σε μεταβλητές περιβάλλοντος `.env`. Για λόγους ευκολίας εγκατάστασης και επίδειξης, το αρχείο θα πρέπει να τροποποιηθεί ώστε αντί για μεταβλητές περιβάλλοντος να χρησιμοποιηθούν απευθείας σταθερές τιμές, όπως φαίνεται παρακάτω:

```
4   const pool = new Pool({
5       user: 'postgres',
6       host: 'localhost',
7       database: 'automata_db',
8       password: '1234',
9       port: 5432,
10  });
11  module.exports = pool;
```

Εικόνα 13. Παράδειγμα σταθερών τιμών στο αρχείο database.js

5.1.4 Εκκίνηση Διακομιστή

Αρχικά, ανοίγουμε το τερματικό και μεταφερόμαστε στον φάκελο backend της εφαρμογής:

```
cd finite-automata-web\backend
```

Έπειτα, ανοίγουμε τον διακομιστή Node.js με την εντολή :

```
node server.js
```

Αν ο διακομιστής έχει ξεκινήσει επιτυχώς, τότε θα είναι πλέον διαθέσιμος στη διεύθυνση http://localhost:3000, η οποία θα εμφανίζει κατάλληλο μήνυμα επιβεβαίωσης.

5.1.5 Εκκίνηση Εφαρμογής

Η διεπαφή χρήστη (frontend) είναι στατικά αρχεία HTML, CSS και JavaScript και μπορεί να εκτελεστεί απευθείας μέσω browser. Ανοίξτε το αρχείο index.html με διπλό κλικ μέσω ενός φυλλομετρητή (browser) ή μέσω της επέκτασης Live Server (αν χρησιμοποιείτε VS Code), ώστε να ενεργοποιηθεί η εφαρμογή .

Με την ολοκλήρωση των παραπάνω βημάτων, η εφαρμογή είναι πλήρως λειτουργική σε τοπικό περιβάλλον. Ο χρήστης έχει πλέον τη δυνατότητα να σχεδιάζει αυτόματα, να αποθηκεύει και να φορτώνει αυτά, είτε τοπικά είτε μέσω της βάσης, καθώς και να εκτελεί προσομοιώσεις αποδοχής συμβολοσειρών σε πραγματικό χρόνο.

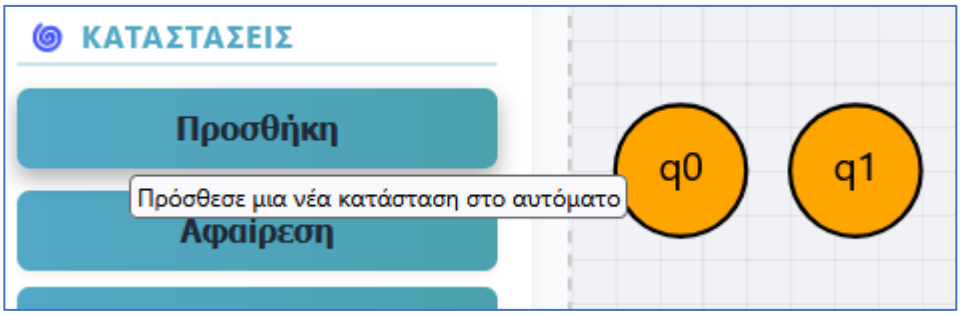
5.2 Οδηγός Χρήσης Εφαρμογής

Η εφαρμογή σχεδιάστηκε ώστε να είναι φιλική προς τον χρήστη και να επιτρέπει εύκολη δημιουργία, επεξεργασία και προσομοίωση πεπερασμένων αυτομάτων. Ακολουθεί μια αναλυτική περιγραφή των βασικών λειτουργιών και του τρόπου χρήσης της.

Εικόνα 14. Διεπαφή της εφαρμογής

5.2.1 Καταστάσεις

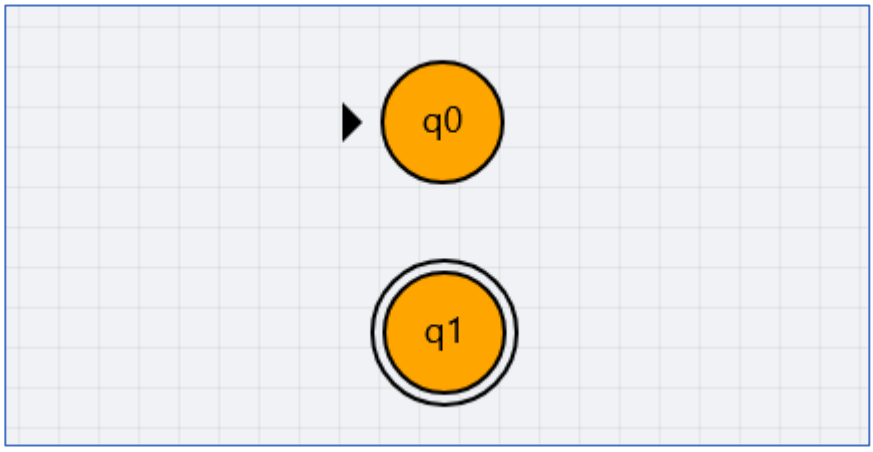
Στο πλευρικό μενού, που βρίσκεται στην αριστερή πλευρά της οθόνης, υπάρχουν διαθέσιμες επιλογές για τη διαχείριση των καταστάσεων, των μεταβάσεων καθώς και για αποθήκευση ή φόρτωση αυτομάτων. Ο χρήστης μπορεί να προσθέσει νέες καταστάσεις επιλέγοντας το κουμπί «Προσθήκη» από το μενού. Κάθε νέα κατάσταση εμφανίζεται στον καμβά (SVG area) και μπορεί να μετακινηθεί ελεύθερα μεταξύ των ορίων, μέσω λειτουργίας drag-and-drop.



Εικόνα 15. Προσθήκη νέων καταστάσεων στον καμβά

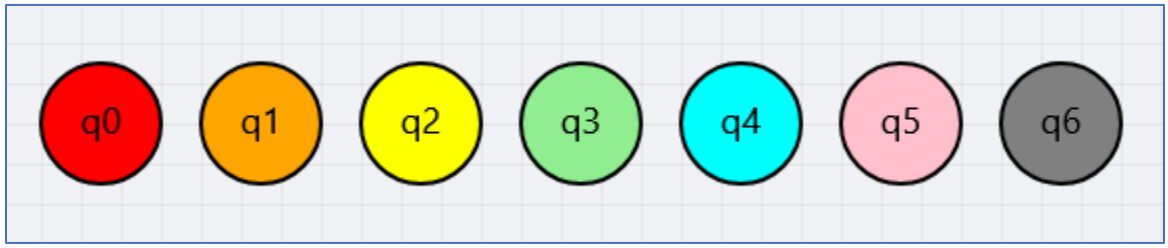
Για να οριστεί μία κατάσταση ως αρχική, ο χρήστης επιλέγει πρώτα την επιθυμητή κατάσταση και στη συνέχεια θα πατήσει το κουμπί «Ορισμός Αρχικής». Οπτικά, η αρχική κατάσταση επισημαίνεται με ένα βέλος που υποδεικνύει την αρχή του αυτομάτου, καθιστώντας την άμεσα αναγνωρίσιμη. Αντίστοιχα, για τον ορισμό τελικών καταστάσεων, ο χρήστης επιλέγει την κατάσταση και πατάει το κουμπί

«Ορισμός Τελικής», το οποίο θα δημιουργήσει έναν δεύτερο κύκλο εξωτερικά της κατάστασης. Σημειώνεται ότι κάθε ΠΑ διαθέτει μία και μόνο μία αρχική κατάσταση, ενώ ταυτόχρονα μπορεί να έχει πολλαπλές τελικές καταστάσεις, εφόσον υπάρχουν περισσότερες από μία διαδρομές αποδοχής συμβολοσειρών.



Εικόνα 16. Παράδειγμα εμφάνισης αρχικής και τελικής κατάστασης

Με παρόμοιο τρόπο λειτουργούν και οι υπόλοιπες επιλογές του μενού για τη διαχείριση των καταστάσεων. Για παράδειγμα, για την αφαίρεση μιας κατάστασης, ο χρήστης επιλέγει πρώτα την κατάσταση για διαγραφή και έπειτα πατάει το κουμπί «Αφαίρεση». Επιπλέον, υπάρχει η δυνατότητα αλλαγής χρώματος μιας κατάστασης επιλέγοντας από μια προεπιλεγμένη παλέτα βασικών, ευδιάκριτων χρωμάτων.

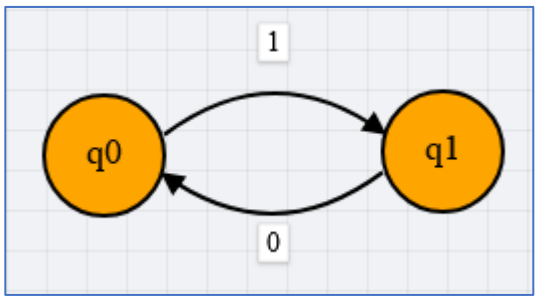


Εικόνα 17. Διαθέσιμη χρωματική παλέτα για τις καταστάσεις

5.2.2 Μεταβάσεις

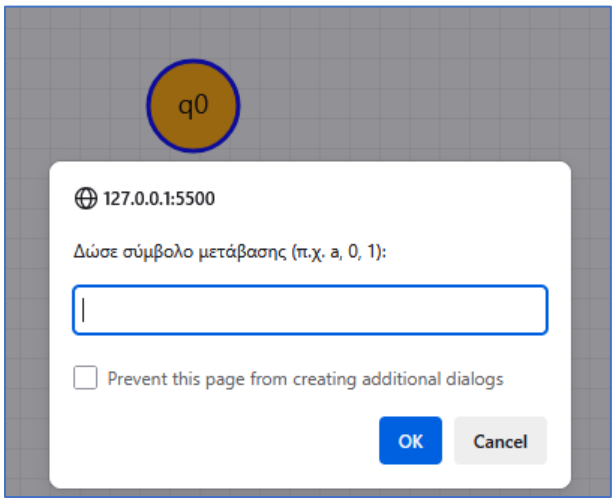
Με ανάλογο τρόπο λειτουργεί και η δημιουργία μεταβάσεων μεταξύ των καταστάσεων. Ο χρήστης πατάει το κουμπί «Προσθήκη» που αναφέρεται στις μεταβάσεις και επιλέγει πρώτα την κατάσταση από την οποία θα ξεκινάει η μετάβαση και έπειτα την κατάσταση στην οποία θα καταλήγει. Θα εμφανιστεί ένα

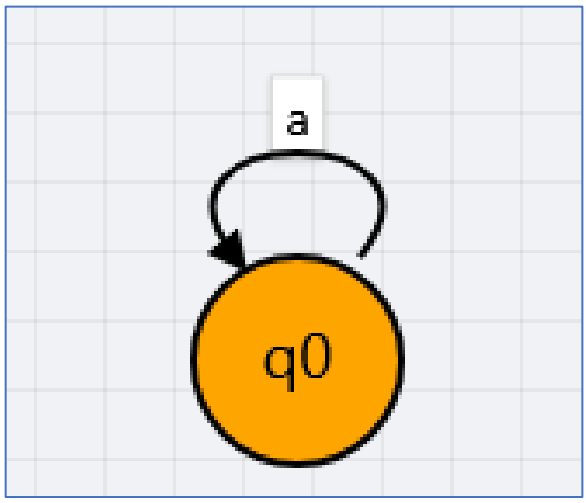
πεδίο όπου ο χρήστης εισάγει το σύμβολο εισόδου που αντιστοιχεί στη μετάβαση. Στην περίπτωση μη-ντετερμινιστικού αυτομάτου, επιτρέπεται η προσθήκη πολλαπλών μεταβάσεων για το ίδιο σύμβολο αρχίζοντας από την ίδια κατάσταση, όπως και η προσθήκη μεταβάσεων με το κενό σύμβολο (ε). Αντίθετα, στα ντετερμινιστικά αυτόματα, η εφαρμογή απαγορεύει την μετάβαση με κενό σύμβολο και περιορίζει τον χρήστη ώστε να υπάρχει ακριβώς μία μετάβαση για κάθε κατάσταση και σύμβολο εισόδου.



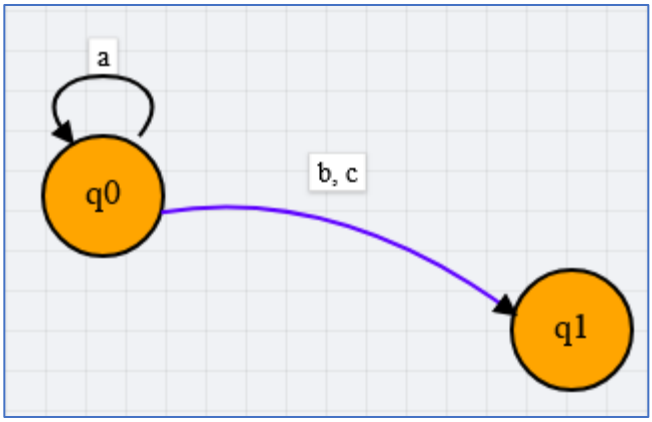
Εικόνα 18. Παράδειγμα εμφάνισης μεταβάσεων μεταξύ των καταστάσεων

Η εφαρμογή επιτρέπει επίσης τη δημιουργία επαναληπτικού βρόχου (self-loop) στην ίδια κατάσταση, μέσω της επιλογής «Επαναληπτικός βρόχος» από το μενού. Ο χρήστης επιλέγει την κατάσταση, εισάγει το σύμβολο εισόδου στο σχετικό πεδίο και δημιουργείται ένα βέλος το οποίο ξεκινά και καταλήγει στην ίδια κατάσταση.





Οι μεταβάσεις μπορούν να επιλεχθούν από τον χρήστη, οπότε και επισημαίνονται με μωβ χρώμα για όσο παραμένουν επιλεγμένες, διευκολύνοντας την οπτική αναγνώριση. Στη συνέχεια, με την επιλογή «Επεξεργασία», ο χρήστης μπορεί να τροποποιήσει το σύμβολο εισόδου της επιλεγμένης μετάβασης ή ακόμα και να προσθέσει περισσότερα σύμβολα διαχωρισμένα με κόμμα.

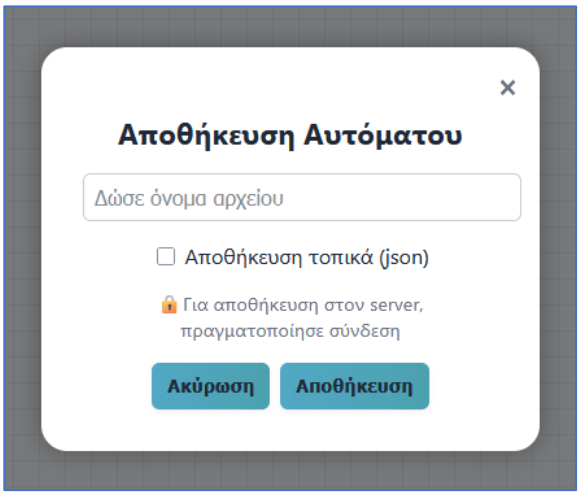


Εικόνα 19. Εμφάνιση επιλεγμένης μετάβασης από τον χρήστη

Τέλος, η επιλογή «Αφαίρεση» επιτρέπει τη διαγραφή της επιλεγμένης μετάβασης από το αυτόματο.

5.2.3 Αποθήκευση & Φόρτωση

Η εφαρμογή παρέχει στον χρήστη τη δυνατότητα αποθήκευσης του σχεδιασμένου αυτομάτου, είτε τοπικά στον υπολογιστή του, είτε στον server, εφόσον ο χρήστης είναι συνδεδεμένος στον λογαριασμό του. Η επιλογή «Αποθήκευση» , που εμφανίζεται στο πλευρικό μενού, ανοίγει ένα αναδυόμενο παράθυρο (modal) που προσφέρει στον χρήστη διάφορες επιλογές αποθήκευσης.



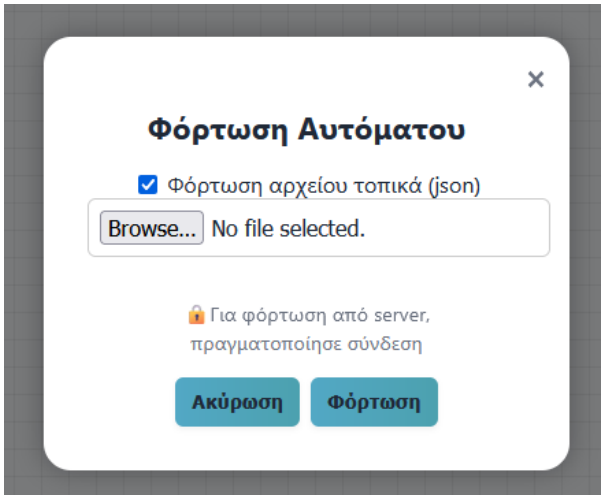
Εικόνα 20. Modal με επιλογές αποθήκευσης για guest user

Στο παράθυρο αυτό, ο χρήστης καλείται να εισάγει ένα όνομα για το αρχείο στο αντίστοιχο πεδίο και επιλέγει αν επιθυμεί τοπική αποθήκευση, διαλέγοντας το σχετικό πλαίσιο επιλογής (checkbox). Εάν επιλέξει την τοπική αποθήκευση, το

σχεδιασμένο αυτόματο αποθηκεύεται ως αρχείο τύπου JSON στον υπολογιστή του χρήστη, το οποίο μπορεί να χρησιμοποιηθεί αργότερα για φόρτωση ή επεξεργασία.

Εάν ο χρήστης δεν επιλέξει τοπική αποθήκευση και είναι συνδεδεμένος στον λογαριασμό του, το αυτόματο αποθηκεύεται στον server. Η επιλογή για αποθήκευση στον server είναι κλειδωμένη, σε περίπτωση που ο χρήστης δεν έχει πραγματοποιήσει σύνδεση στον λογαριασμό του. Τα αυτόματα που αποθηκεύονται στον server, είναι επίσης μορφής JSON και είναι προσβάσιμα μόνο από τον λογαριασμό χρήστη που χρησιμοποιήθηκε για την αποθήκευση του.

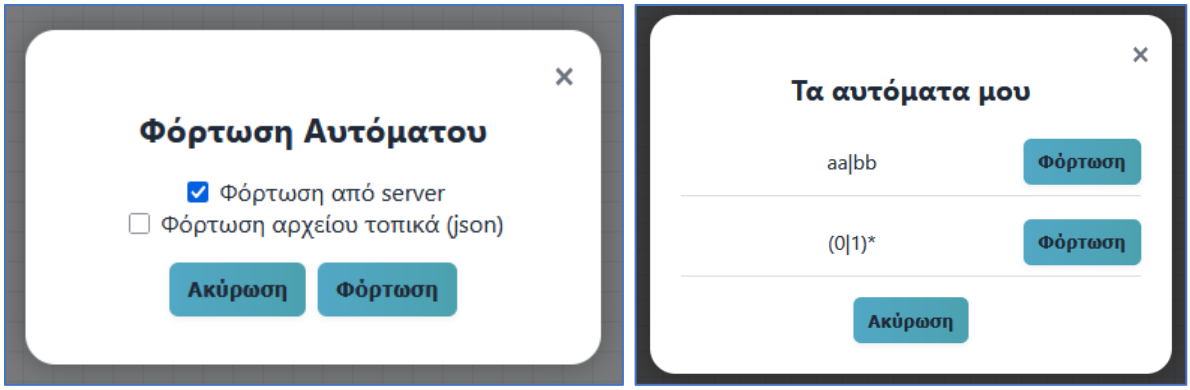
Αντίστοιχα, η επιλογή «Φόρτωση» στο πλευρικό μενού ανοίγει ένα παρόμοιο παράθυρο, το οποίο επιτρέπει στον χρήστη να επιλέξει τρόπο φόρτωσης αρχείου. Ο χρήστης μπορεί να επιλέξει είτε φόρτωση τοπικού αρχείου JSON, είτε φόρτωση αυτομάτου από τον server, εφόσον είναι συνδεδεμένος στον λογαριασμό του.



Εικόνα 21. Modal με επιλογές φόρτωσης για guest user

Με την επιλογή τοπικής φόρτωσης ανοίγει η εξερεύνηση αρχείων του λειτουργικού συστήματος του υπολογιστή, επιτρέποντας στον χρήστη να διαλέξει το αρχείο που επιθυμεί να φορτώσει στην εφαρμογή. Η εφαρμογή δέχεται αρχεία τύπου JSON, τα οποία έχουν δημιουργηθεί από προηγούμενη αποθήκευση του αυτομάτου μέσω της ίδιας εφαρμογής.

Στην περίπτωση φόρτωσης από τον server, εμφανίζεται η λίστα με όλα τα αποθηκευμένα αυτόματα που σχετίζονται με τον λογαριασμό του χρήστη στη βάση δεδομένων. Ο χρήστης μπορεί να επιλέξει ποιο από τα αυτόματα της λίστας επιθυμεί να φορτώσει.



Με την ολοκλήρωση της λειτουργίας της φόρτωσης, το αυτόματο εμφανίζεται στον καμβά της εφαρμογής, διατηρώντας τη θέση των καταστάσεων, τα χρώματα και τις μεταβάσεις ακριβώς όπως είχαν αποθηκευτεί.

Τα αυτόματα που αποθηκεύονται, είτε βρίσκονται τοπικά, είτε στον server, είναι σε αρχεία της μορφής JSON (JavaScript Object Notation). Στην παρούσα εφαρμογή, το JSON περιέχει πληροφορίες για τον τύπο του αυτομάτου, για τις καταστάσεις, αν είναι αρχικές ή τελικές, το χρώμα τους και τις συντεταγμένες τους στον καμβά SVG, καθώς και για τις μεταβάσεις και τα σύμβολα εισόδου τους. Η δομή του είναι προσαρμοσμένη για χρήση στον καμβά SVG της συγκεκριμένης εφαρμογής.

```
{
  "type": "DFA",
  "states": [
    {"id": "q0","x": 181,"y": 128,"color": "orange","isInitial": true,"isFinal": false},
    {"id": "q1","x": 455,"y": 126,"color": "orange","isInitial": false,"isFinal": true}
  ],
  "transitions": [
    {"from": "q0","to": "q0","symbol": "0"},
    {"from": "q1","to": "q1","symbol": "1"},
    {"from": "q0","to": "q1","symbol": "1"},
    {"from": "q1","to": "q0","symbol": "0"}
  ]
}
```

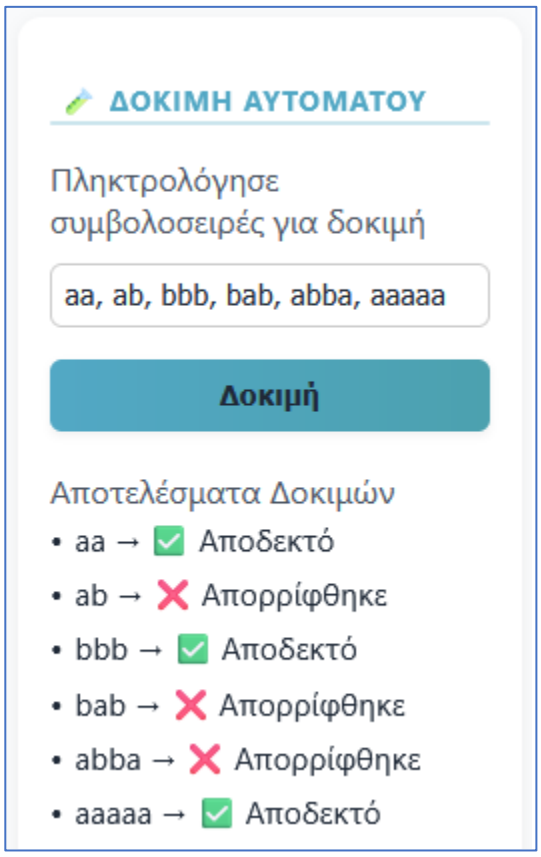
Εικόνα 22. DFA αποθηκευμένο σε αρχείο JSON

5.2.4 Έλεγχος Συμβολοσειράς

Στο δεξί πλευρικό μενού, παρέχεται η δυνατότητα ελέγχου συμβολοσειρών εισόδου ως προς την αποδοχή ή απόρριψη τους από το αυτόματο που έχει σχεδιαστεί. Ο χρήστης εισάγει τη συμβολοσειρά που επιθυμεί να ελέγξει στο αντίστοιχο πεδίο και πατάει το κουμπί «Δοκιμή».

Η εφαρμογή εκτελεί τη διαδικασία προσομοίωσης και ελέγχει τη συμβολοσειρά ανά σύμβολο στο ΠΑ. Στην περίπτωση ντετερμινιστικού αυτομάτου, η εφαρμογή ακολουθεί τη μοναδική διαδρομή που καθορίζεται από τις μεταβάσεις του αυτομάτου. Στην περίπτωση μη-ντετερμινιστικού αυτομάτου, εξετάζονται όλες οι δυνατές διαδρομές παράλληλα, λαμβάνοντας υπόψη τις πιθανές εναλλακτικές μεταβάσεις, καθώς και τις μεταβάσεις με κενό σύμβολο (ε).

Η προσομοίωση καταλήγει με μήνυμα ενημέρωσης προς τον χρήστη, το οποίο δηλώνει αν η συμβολοσειρά έγινε αποδεκτή (δηλαδή αν κατέληξε σε τελική κατάσταση) ή αν έχει απορριφθεί (δηλαδή αν δεν κατέληξε σε καμία τελική κατάσταση).

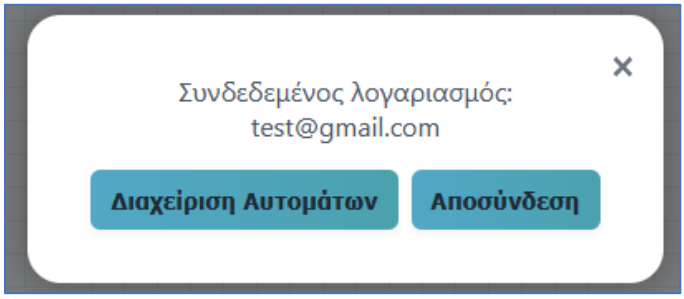


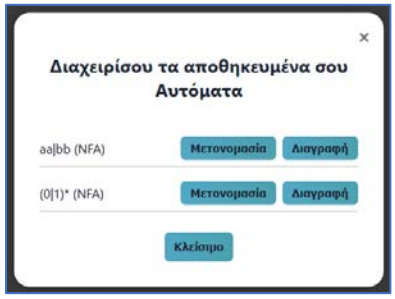
Εικόνα 23. Παράδειγμα του μενού για έλεγχο συμβολοσειρών

5.2.5 Περαιτέρω λειτουργίες

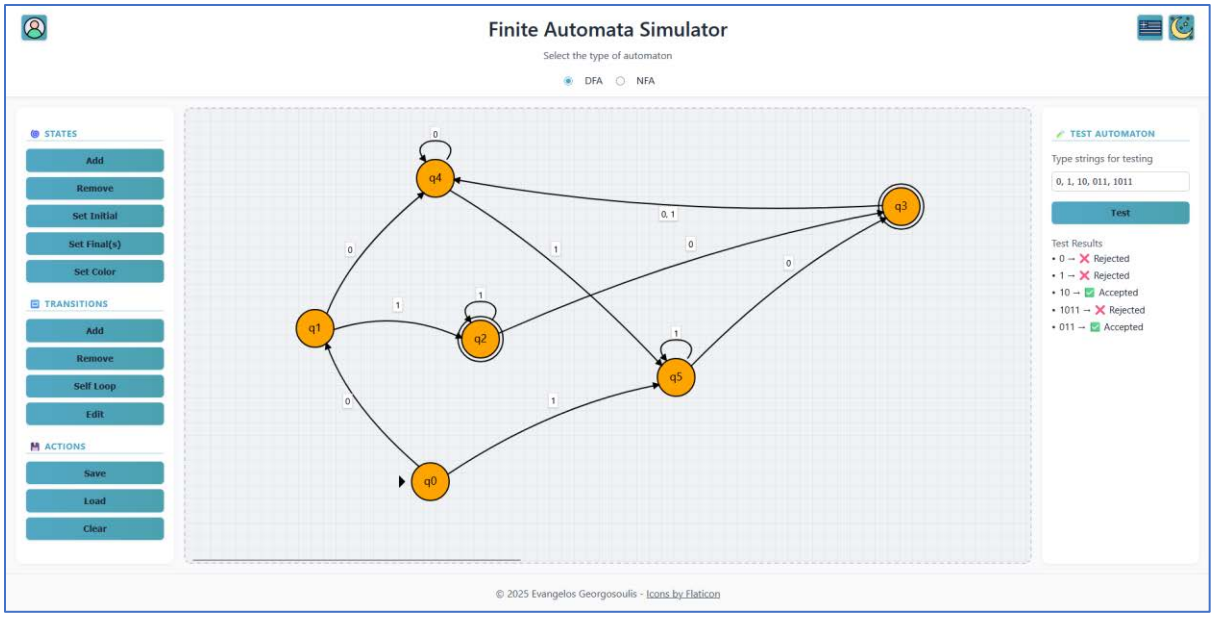
Η εφαρμογή παρέχει επιπλέον λειτουργίες για την καλύτερη εμπειρία χρήστη, οι οποίες ενισχύουν την ευχρηστία και την προσαρμοστικότητα της διεπαφής.

Δίνεται η δυνατότητα δημιουργίας λογαριασμού μέσω της επιλογής πάνω αριστερά, όπου ο χρήστης εισάγει τα στοιχεία του (email και κωδικό πρόσβασης). Στη συνέχεια, ο χρήστης μπορεί να πραγματοποιήσει σύνδεση με τα στοιχεία του, αποκτώντας έτσι πρόσβαση σε λειτουργίες όπως η αποθήκευση και φόρτωση αυτομάτων στον server. Μετά τη σύνδεση, στην εφαρμογή εμφανίζεται πληροφοριακό παράθυρο με τα στοιχεία του χρήστη, ενώ παρέχονται οι επιλογές διαχείρισης αυτομάτων και αποσύνδεσης. Η επιλογή «Διαχείριση Αυτομάτων» ανοίγει νέο παράθυρο που εμφανίζει τη λίστα με τα αποθηκευμένα αυτόματα του λογαριασμού (εάν υπάρχουν). Από το παράθυρο αυτό, ο χρήστης έχει τη δυνατότητα να πραγματοποιήσει μετονομασία ενός αυτομάτου ή να επιλέξει τη διαγραφή του από τον server. Οι αλλαγές αυτές ενημερώνουν αυτόματα τη βάση δεδομένων και εμφανίζονται άμεσα στη λίστα.



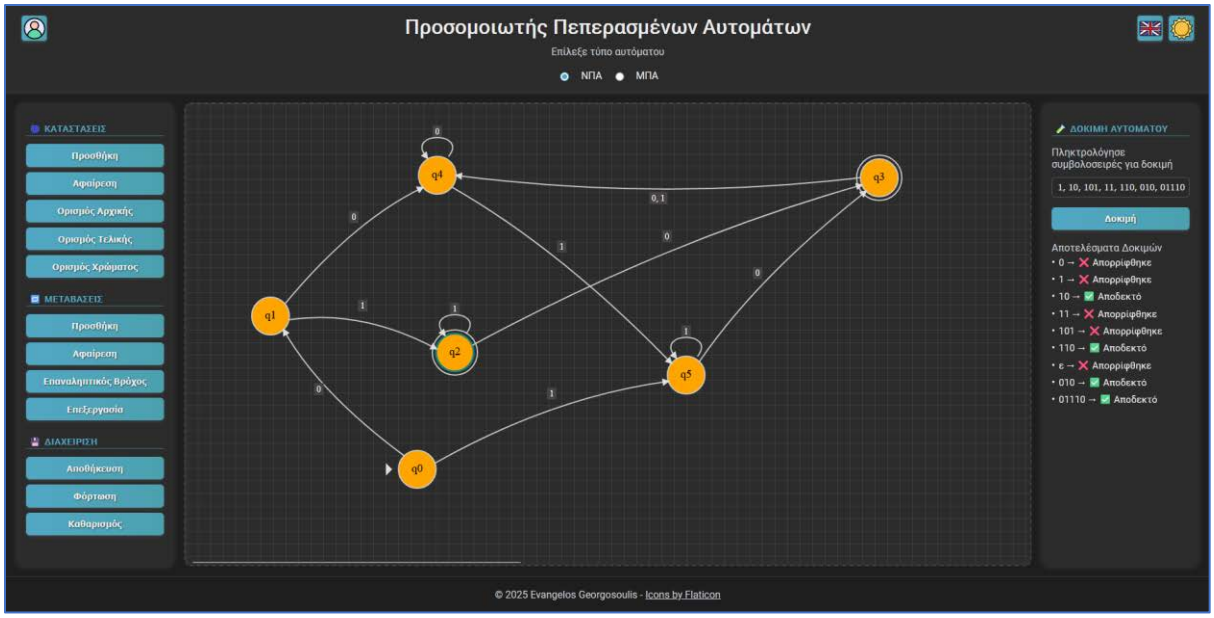


Η εφαρμογή υποστηρίζει επίσης πολυγλωσσικό περιβάλλον με δυνατότητα εναλλαγής μεταξύ Ελληνικής και Αγγλικής γλώσσας. Ο χρήστης μπορεί να αλλάξει γλώσσα οποιαδήποτε στιγμή μέσω της επιλογής αλλαγής γλώσσας, με αποτέλεσμα να μεταφράζονται δυναμικά όλα τα κείμενα, τα μηνύματα και τα μενού της εφαρμογής.



Εικόνα 24. Εμφάνιση εφαρμογής με γλώσσα εμφάνισης τα Αγγλικά (UK)

Δίπλα από την επιλογή γλώσσας, βρίσκεται η λειτουργία αλλαγής θέματος, με την οποία ο χρήστης μπορεί να επιλέξει ανάμεσα σε σκοτεινό (dark mode) και φωτεινό (light mode) περιβάλλον. Η εναλλαγή του θέματος επηρεάζει όλη την εμφάνιση της εφαρμογής, συμπεριλαμβανομένου του καμβά, των αυτομάτων, των μενού και των διαλόγων, ώστε να είναι πιο ευχάριστη και ξεκούραστη η χρήση της εφαρμογής ανάλογα με τις συνθήκες φωτισμού.



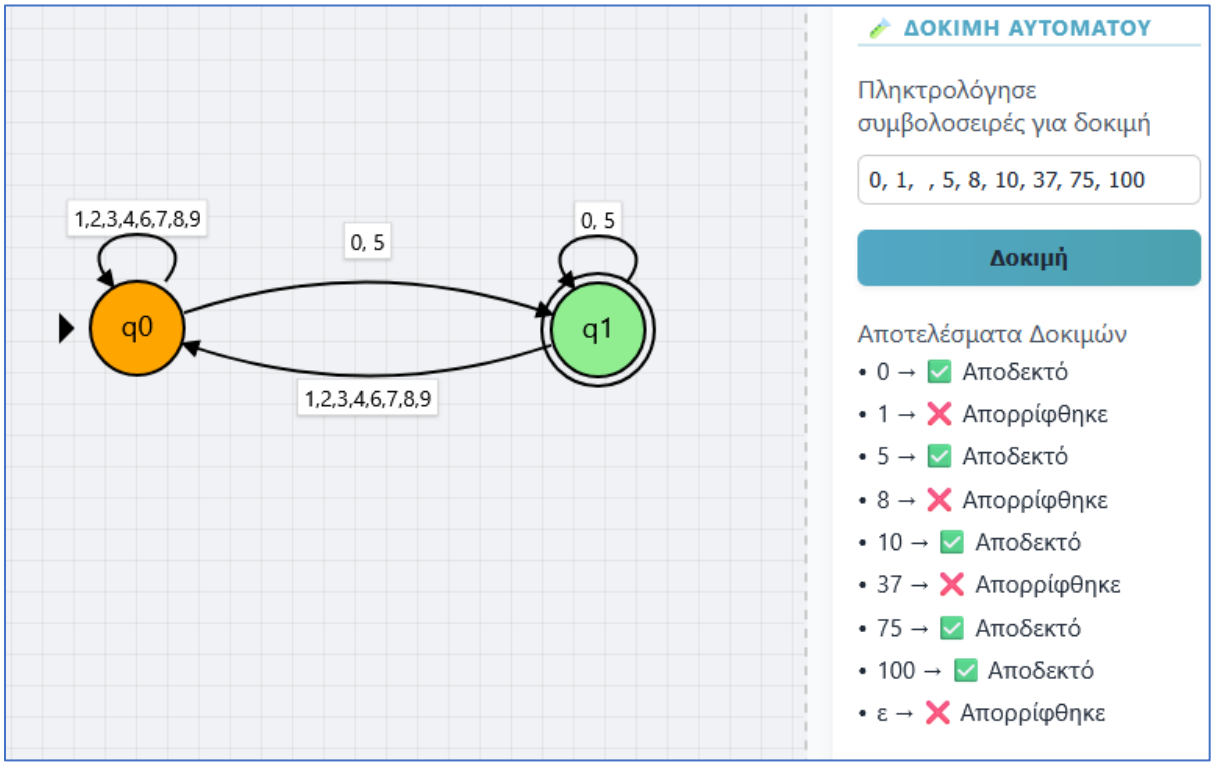
Εικόνα 25. Εμφάνιση εφαρμογής σε σκοτεινό θέμα

Μέσω της επιλογής «Καθαρισμός» που βρίσκεται κάτω από τις επιλογές της αποθήκευσης και φόρτωσης, ο χρήστης μπορεί να διαγράψει όλες τις καταστάσεις, μεταβάσεις και ιδιότητες που έχουν σχεδιαστεί στον καμβά, ώστε να ξεκινήσει ένα νέο αυτόματο. Πριν την ολοκλήρωση της ενέργειας, εμφανίζεται προειδοποιητικό μήνυμα επιβεβαίωσης για την αποφυγή τυχόν απώλειας δεδομένων.

5.3 Παραδείγματα Σχεδίασης Αυτομάτων

5.3.1 Παράδειγμα 1 – Πολλαπλάσια του 5 (DFA)

Στην παρακάτω εικόνα παρουσιάζεται ένα ντετερμινιστικό ΠΑ (DFA) το οποίο δέχεται ακέραια ψηφία από το 0 έως το 9 και ελέγχει αν είναι πολλαπλάσια του 5.



Εικόνα 26. DFA που ελέγχει αν ένας αριθμός είναι πολλαπλάσιος του 5.

Το αυτόματο αποτελείται από δύο καταστάσεις:

- Την q0 (αρχική κατάσταση)
- Την q1 (τελική κατάσταση - αποδοχής)

Στην αρχική κατάσταση q0 υπάρχει επαναληπτική μετάβαση στην q0 για τα ψηφία 1,2,3,4,6,7,8,9 ενώ για το 0 και το 5 γίνεται μετάβαση προς την q1. Η τελική

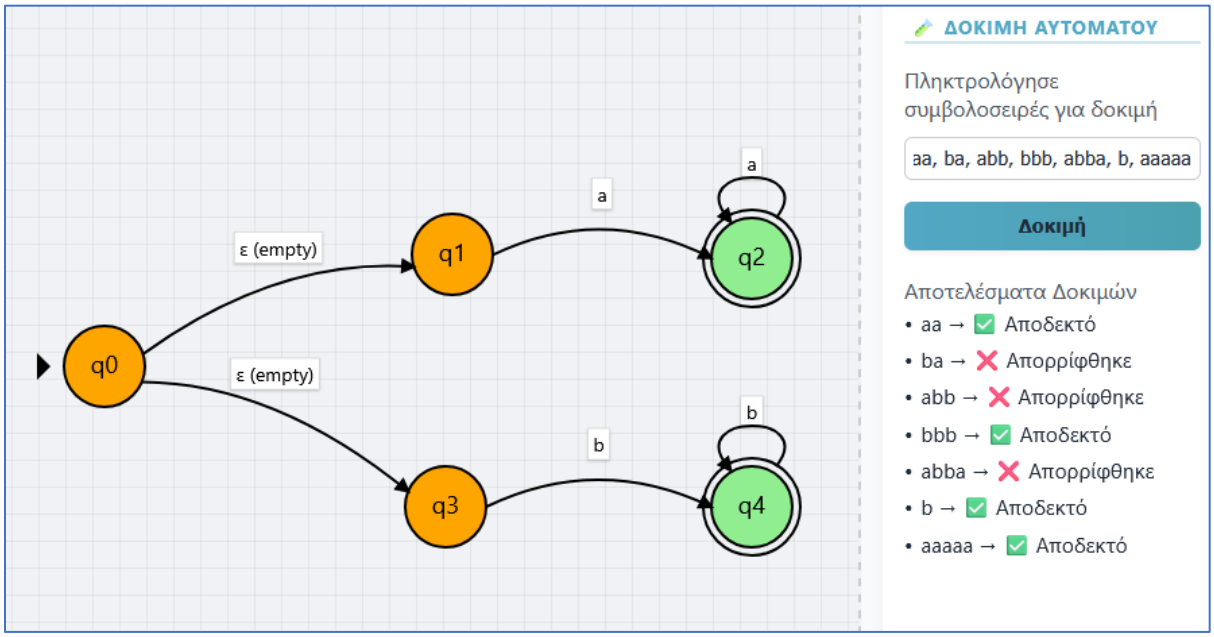
κατάσταση q1 έχει επαναληπτική μετάβαση στον εαυτό της για τα 0 και 5, επιτρέποντας την αποδοχή πολλαπλών εισόδων που είναι διαδοχικά πολλαπλάσια του 5 και μια μετάβαση προς την q0 για τα υπόλοιπα ψηφία.

Το αυτόματο βασίζεται στην ιδιότητα ότι ένας αριθμός είναι πολλαπλάσιος του 5 αν και μόνο αν το τελευταίο ψηφίο του είναι 0 ή 5. Επομένως, αρκεί να ελέγχεται το τελευταίο ψηφίο της συμβολοσειράς, χωρίς να λαμβάνονται υπόψη τα προηγούμενα ψηφία. Με αυτόν τον τρόπο, η σχεδίαση του αυτομάτου απλοποιείται σε δύο καταστάσεις, μία αρχική και μία τελική, με μεταβάσεις που καθορίζονται μόνο από το τελευταίο τρέχον ψηφίο.

Στη δεξιά πλευρά της εφαρμογής εμφανίζονται τα αποτελέσματα δοκιμών για διαφορετικές συμβολοσειρές εισόδου. Παρατηρούμε ότι οι συμβολοσειρές “0”, “5”, “10”, “75” και “100” έγιναν αποδεκτές, ενώ οι “1”, “8”, “37” και “ε (κενό)” απορρίφθηκαν επιβεβαιώνοντας τη σωστή λειτουργία του αυτομάτου.

5.3.2 Παράδειγμα 2 – Συμβολοσειρές ιδίου συμβόλου (NFA)

Στην εικόνα 27 παρουσιάζεται ένα μη-ντετερμινιστικό ΠΑ (NFA) το οποίο αποδέχεται συμβολοσειρές που περιέχουν τα ίδια σύμβολα, μεταξύ a και b.



Εικόνα 27. NFA που ελέγχει αν η συμβολοσειρά αποτελείται αποκλειστικά από το ίδιο σύμβολο

Το αυτόματο αποτελείται από 5 καταστάσεις

- q0 (αρχική κατάσταση)
- q1, q3 (ενδιάμεσες καταστάσεις)
- q2, q4 (τελικές καταστάσεις)

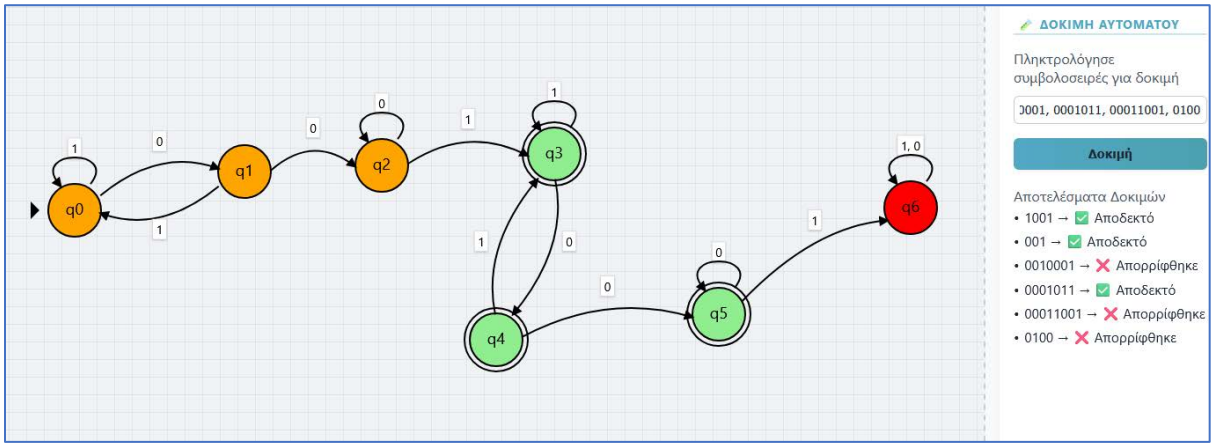
Από την αρχική κατάσταση q0 υπάρχουν δύο έψιλον-μεταβάσεις: μία προς την q1 και μία προς την q3, επιτρέποντας στο αυτόματο να ξεκινήσει την αναζήτηση είτε για "a" είτε για "b" χωρίς να διαβάσει σύμβολο εισόδου.

Από την q1 υπάρχει μετάβαση με σύμβολο "a" προς την τελική κατάσταση q2, ενώ από την q3 υπάρχει μετάβαση με σύμβολο "b" προς την τελική q4. Στις τελικές καταστάσεις υπάρχουν επαναληπτικές μεταβάσεις στον εαυτό τους με το ίδιο σύμβολο (a ή b αντίστοιχα), επιτρέποντας την αποδοχή οποιασδήποτε συμβολοσειράς που συνεχίζει με τα ίδια σύμβολα.

Στη δεξιά πλευρά εμφανίζονται τα αποτελέσματα της δοκιμής για τις δοθέντες συμβολοσειρές. Συγκεκριμένα, οι "aa", "bbb", "b" και "aaaaa" έγιναν αποδεκτές, ενώ οι "ba", "abb" και "abba" απορρίφθηκαν. Επομένως, το αυτόματο αναγνωρίζει επιτυχώς αν η συμβολοσειρά αποτελείται μόνο από τους χαρακτήρες "a" ή μόνο από τους "b".

5.3.3 Παράδειγμα 3 – Εμφάνιση 001 μία φορά (DFA)

Στην παρακάτω εικόνα παρουσιάζεται ένα ντετερμινιστικό ΠΑ (DFA) το οποίο δέχεται δυαδικές συμβολοσειρές (αποτελούμενες από τα σύμβολα 0 και 1) που περιέχουν ακριβώς μία φορά την ακολουθία 001 και απορρίπτει αυτές που εμφανίζουν την ακολουθία παραπάνω φορές.



Εικόνα 28. DFA που δέχεται συμβολοσειρές με ακριβώς μία εμφάνιση της ακολουθίας "001"

Το αυτόματο αποτελείται από 7 καταστάσεις:

- q0: αρχική κατάσταση
- q1: έχει διαβαστεί 0
- q2: έχει διαβαστεί 00
- q3: έχει διαβαστεί 001 (κατάσταση αποδοχής)
- q4: μετά την εμφάνιση της ακολουθίας 001, επιτρέπεται οποιοδήποτε σύμβολο 0 ή 1
- q5: έχει διαβαστεί 001 μία φορά και έχει διαβαστεί ακόμα 00
- q6: κατάσταση απόρριψης (έχει διαβαστεί 001 και δεύτερη φορά)

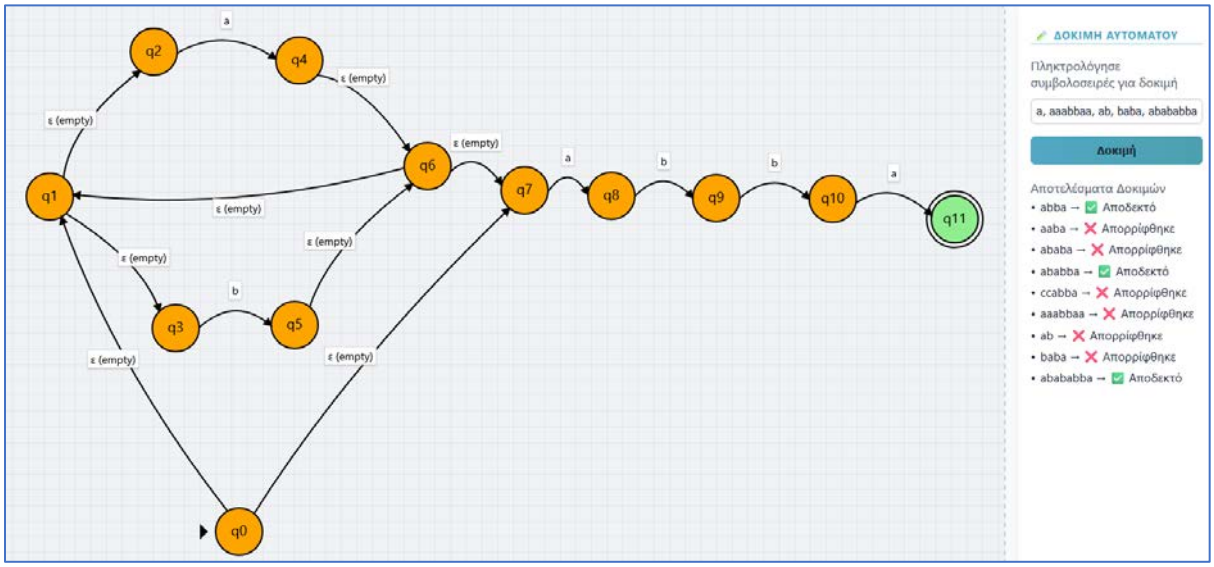
Από την αρχική κατάσταση q0, το αυτόματο μεταβαίνει στην q1 με ανάγνωση ενός 0 και στη συνέχεια στην q2 με ένα δεύτερο 0. Μόλις διαβαστεί 1, το αυτόματο μεταβαίνει στην τελική κατάσταση q3, σηματοδοτώντας ότι η ακολουθία έχει εντοπιστεί για πρώτη φορά.

Από την q3 επιτρέπεται η ανάγνωση οποιουδήποτε συμβόλου 0 ή 1. Με 1 παραμένει στην τρέχουσα κατάσταση αλλά με 0 συνεχίζει στην q4, η οποία είναι και αυτήν τελική. Στην q4 άμα διαβαστεί ξανά 0 τότε απομένει να διαβαστεί το 1 για να σχηματιστεί η ακολουθία 001 και δεύτερη φορά. Όσο γίνεται ανάγνωση του 0 το αυτόματο παραμένει στην q5, όμως με το που αναγνωστεί το 1 τότε μεταβαίνει στην q6. Η q6 θεωρείται κατάσταση απόρριψης, γιατί πλέον έχει εμφανιστεί δύο φορές η ακολουθία 001 οπότε από εδώ και στο εξής ότι σύμβολο και να διαβαστεί, η συμβολοσειρά θα απορριφθεί.

Στο μενού των δοκιμών εμφανίζονται και τα αποτελέσματα για διάφορες συμβολοσειρές. Παρατηρούμε ότι οι συμβολοσειρές “001”, “1001”, “0001011” έγιναν αποδεκτές, καθώς περιέχουν ακριβώς μία φορά την ακολουθία 001. Αντίθετα, οι “0010001”, “00011001” και “0100” απορρίφθηκαν, επιβεβαιώνοντας ότι το αυτόματο λειτουργεί σωστά, αποδεχόμενο μόνο τις συμβολοσειρές με μοναδική εμφάνιση της ακολουθίας 001.

5.3.4 Παράδειγμα 4 – Επίθεμα “abba” (NFA)

Η παρακάτω εικόνα παρουσιάζει ένα μη-ντετερμινιστικό πεπερασμένο αυτόματο (NFA), το οποίο αποδέχεται συμβολοσειρές που καταλήγουν στην ακολουθία “abba”. Το αυτόματο επιτρέπει οποιονδήποτε συνδυασμό από τα γράμματα a και b πριν από αυτήν την ακολουθία, αρκεί η συμβολοσειρά να τελειώνει με “abba”. Ουσιαστικά αναγνωρίζει τη γλώσσα που περιγράφεται από την Κανονική Έκφραση: (a | b)* abba. Οι ΚΕ χρησιμοποιούνται για τον ορισμό συνόλων συμβολοσειρών με έναν σύντομο τρόπο και μπορούν να αναπαρασταθούν με αυτόματα όπως το παρακάτω, που ελέγχουν αν μια συμβολοσειρά ανήκει στη γλώσσα αυτή.



Εικόνα 29. NFA που δέχεται συμβολοσειρές με το επίθεμα “abba”

Το αυτόματο αποτελείται από 12 καταστάσεις, ξεκινώντας από την αρχική q0. Μέσω ε-μεταβάσεων, μεταβαίνει σε διαδρομές που αναζητούν σταδιακά την εμφάνιση των χαρακτήρων a, b, b, a στη σωστή σειρά. Μόλις διαβαστούν διαδοχικά τα a, b, b, a το αυτόματο φτάνει στην τελική κατάσταση q11, όπου αποδέχεται τη συμβολοσειρά. Εάν μετά την αναγνώριση της ακολουθίας διαβαστεί οποιοδήποτε επιπλέον σύμβολο, η συμβολοσειρά απορρίπτεται.

Δεξιά, εμφανίζονται τα αποτελέσματα δοκιμής για διαφορετικές συμβολοσειρές. Για παράδειγμα οι “abba”, “ababba”, “abababba” γίνονται αποδεκτές, αφού τελειώνουν με abba, ενώ συμβολοσειρές όπως “aaba” και “abab” απορρίπτονται, επειδή δεν πληρούν το κριτήριο.

6. ΕΠΙΛΟΓΟΣ

6.1 Συμπεράσματα

Η παρούσα πτυχιακή εργασία είχε ως στόχο την ανάπτυξη μιας διαδραστικής διαδικτυακής εφαρμογής για τη σχεδίαση, ανάλυση και προσομοίωση Πεπερασμένων Αυτομάτων (DFA & NFA). Μέσα από τη μελέτη του θεωρητικού υποβάθρου, τη σύγκριση υπαρχόντων εργαλείων και την ανάλυση των απαιτήσεων, σχεδιάστηκε και υλοποιήθηκε μια φιλική προς τον χρήστη εφαρμογή με πρακτικές δυνατότητες σχεδίασης, αποθήκευσης, φόρτωσης και δοκιμής αυτομάτων.

Η εφαρμογή αξιοποιεί σύγχρονες τεχνολογίες διαδικτύου με στόχο να προσφέρει μια ολοκληρωμένη εμπειρία τόσο σε φοιτητές που μελετούν τη Θεωρία Υπολογισμού, όσο και σε εκπαιδευτικούς που αναζητούν ένα αποτελεσματικό εργαλείο διδασκαλίας. Μέσα από τις δυνατότητες προσομοίωσης, την υποστήριξη εναλλαγής γλώσσας, θεματικής εμφάνισης καθώς και την αποθήκευση αυτομάτων τοπικά ή σε server, η εφαρμογή επιδιώκει να καλύψει ανάγκες λειτουργικότητας και ευχρηστίας.

Τα παραδείγματα σχεδίασης που παρουσιάστηκαν, τόσο για DFA όσο και για NFA, επιβεβαιώνουν την ορθότητα και την αποτελεσματικότητα της εφαρμογής. Το εργαλείο αποδεικνύεται ικανό να υποστηρίξει τόσο βασικά όσο και πιο σύνθετα σενάρια χρήσης, με ορθή απεικόνιση και ακριβή προσομοίωση. Αν και οι στόχοι της εργασίας επιτεύχθηκαν σε ικανοποιητικό βαθμό, υπάρχουν πεδία που προσφέρονται για μελλοντική επέκταση και βελτίωση, με σκοπό την περαιτέρω ενίσχυση της εκπαιδευτικής και τεχνολογικής της αξίας.

6.2 Μελλοντικές Επεκτάσεις

Παρόλο που η εφαρμογή καλύπτει ένα ευρύ σύνολο λειτουργιών, υπάρχει σημαντικό περιθώριο για εξέλιξη στο μέλλον. Μία σημαντική επέκταση θα μπορούσε να αφορά την υποστήριξη επιπλέον τύπων αυτομάτων, όπως τα αυτόματα στοίβας (pushdown automata - PDA) ή τις μηχανές Turing (Turing machines),

επεκτείνοντας έτσι τη χρήση της εφαρμογής και σε άλλα πεδία της Θεωρίας Υπολογισμού.

Εξίσου χρήσιμη θα ήταν η υποστήριξη βηματικής προσομοίωσης με animation πάνω στο σχεδιασμένο ΠΑ, που θα επιτρέπει στον χρήστη να παρακολουθεί οπτικά κάθε βήμα της εκτέλεσης ενός αυτομάτου, κατανοώντας ευκολότερα τη διαδικασία αποδοχής ή απόρριψης μιας συμβολοσειράς. Επιπρόσθετα, η πλήρης υποστήριξη responsive σχεδίασης θα καθιστούσε την εφαρμογή κατάλληλη για χρήση και από κινητές συσκευές και tablets, αυξάνοντας έτσι την προσβασιμότητα της.

Τέλος, η ενσωμάτωση εκπαιδευτικού υλικού, όπως έτοιμα παραδείγματα, ασκήσεις ή επεξηγηματικά βήματα, θα ενίσχυε τον διδακτικό χαρακτήρα της εφαρμογής και θα την καθιστούσε ένα ολοκληρωμένο εργαλείο μάθησης για τη Θεωρία Υπολογισμού. Με τέτοιες επεκτάσεις, η εφαρμογή μπορεί να εξελιχθεί σε ένα σύγχρονο, διαδραστικό και δυναμικό εκπαιδευτικό περιβάλλον, που θα καλύπτει ακόμα πιο αποτελεσματικά τις ανάγκες φοιτητών και εκπαιδευτικών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1]. K. Georgouli, *Compilers* [Undergraduate textbook]. Kallipos, Open Academic Editions, 2024. [Online]. Available: https://dx.doi.org/10.57713/kallipos-57

[2]. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques and Tools,* 2nd ed. Boston, MA, USA: Addison-Wesley.

[3]. Χ. Δουληγέρης, Α. Καραλής, Ε. Κοπανάκη, and Ρ. Μαυροπόδη, *Τεχνολογίες και Προγραμματισμός στον Παγκόσμιο Ιστό*, 2η έκδ., 2021.

[4]. J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Automata Theory, Languages and Computation*, 3rd ed. Boston, MA, USA: Pearson, 2006.

[5]. E. Gribkoff, "Applications of Deterministic Finite Automata," ECS 120, University of California, Davis, Spring 2013. [Online]. Available: https://web.cs.ucdavis.edu/~rogaway/classes/120/spring13/

[6]. D. Jagdale, "Finite State Machine in Game Development," *Int. J. of Advanced Research in Science, Communication and Technology (IJARSCT),* Oct. 2021. [Online]. Available: https://www.ijarsct.co.in/Paper2062.pdf

[7]. P. Katsaros, *Θεωρία Υπολογισμού και Εφαρμογές* [Undergraduate textbook]. Kallipos, Open Academic Editions, 2015. [Online]. Available: https://dx.doi.org/10.57713/kallipos-478

[8]. E. Zachos, A. Pagourtzis, and T. Souliou, *Computer Science Foundation* [Undergraduate textbook]. Kallipos, Open Academic Editions, 2015. [Online]. Available: https://dx.doi.org/10.57713/kallipos-491

[9]. J. Duckett, *HTML and CSS: Design and Build Websites*. Indianapolis, IN, USA: Wiley, 2011.

[10]. M. Haverbeke, *Eloquent JavaScript*, 4th ed., 2024. [Online]. Available: https://eloquentjavascript.net/

[11]. M. Casciaro and L. Mammino, *Node.js Design Patterns*, 3rd ed. Birmingham, UK: Packt Publishing, 2020.

[12]. PostgreSQL Global Development Group, "About PostgreSQL", *PostgreSQL*, [Online]. Available: https://www.postgresql.org/about/

[13]. Oracle Corporation, "MySQL: About MySQL," *MySQL*, [Online]. Available: https://www.oracle.com/mysql/what-is-mysql/

[14]. MongoDB, Inc., “What is MongoDB?” *MongoDB*, [Online]. Available: https://www.mongodb.com/resources/basics/databases/json-database

[15]. SQLite, “Appropriate Uses for SQLite,” *SQLite*, [Online]. Available: https://www.sqlite.org/about.html

[16]. Microsoft, “SQL Server,” *Microsoft*, [Online]. Available: https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver16

[17]. A. A. Adegoke, “What is PHP? The PHP Programming Language Meaning Explained,” *freeCodeCamp*, 19-Jul-2022. [Online]. Available: https://www.freecodecamp.org/news/what-is-php-the-php-programming-language-meaning-explained/

[18]. “SVG Graphics Introduction,” *W3Schools*. [Online]. Available: https://www.w3schools.com/graphics/svg_intro.asp

[19]. “Chomsky Hierarchy in Theory of Computation,” *GeeksforGeeks*. [Online]. Available: https://www.geeksforgeeks.org/chomsky-hierarchy-in-theory-of-computation/

[20]. Scott Logic, “Finite State Machines,” *Scott Logic Blog*, Dec. 8, 2020. [Online]. Available: https://blog.scottlogic.com/2020/12/08/finite-state-machines.html

[21]. DFA Simulator. [Online]. Available: https://dfa-simulator.vercel.app/

[22]. AutomataVerse. [Online]. Available: https://www.automataverse.com/simulator/dfa

[23]. Automaton Simulator. [Online]. Available: https://automatonsimulator.com/

[24]. JFLAP. [Online]. Available: https://www.jflap.org/