NEU UniGuide: Campus Navigation Application

Near East University Faculty of Engineering Department of Software Engineering

Project Title: NEU UniGuide - Intelligent Campus Navigation System

Prepared by: Taha Mohamed s Bilal Boulaalam

Student ID: 20222588-20213111

Supervisor: Cliff

Academic Year: 2025-2026

Date: January 2026

ABSTRACT

NEU UniGuide is a web-based campus navigation system that serves to facilitate campus life for students, faculty, and visitors at Near East University. The system tackles those usual difficulties that campus members face within large campus buildings while finding faculty members' offices, accessing academic information, or using campus services effectively.

Developed using the latest web technology stack, including Next.js version 14, React version 18, TypeScript, and Tailwind CSS, the application offers a user-friendly and responsive interface that is accessible on various devices. It has an interactive campus map, doctor s faculty members list, artificial intelligence-powered chatbot helper, academic tools set, along with support for English, Turkish, s Arabic languages.

Based on the DeepSeek model and OpenRouter API, this AI chatbot is capable of smartly responding to all the queries entered by users on campus facilities, programs, transport schedules, and other services provided by the university. A tool like this runs searches instantly while keeping results sorted through smart filters. Browser memory holds your data without slowing things down. The back-end works quietly, built for safety when managing settings. Each part connects smoothly, making tasks feel lighter even during heavy use.

Putting together various inputs worked well. The design made navigation straightforward because clarity mattered most. Building the framework came with room to grow since change happens

requirements, and developing an information system for more than 30,000 students from 143 different countries.

This report contains the entire development cycle of the ‘NEU UniGuide’, which includes ‘Problem Analysis’, ‘System Design’, ‘Implementation’, ‘Testing’, as well as ‘Future Upgrades’.

Key words: campus navigation system, web application, ai chatbot, Next.js, React, University Information System, Interactive Map, multi-language support

TABLE OF CONTENTS

1. Introduction
    a. 1.1 Background
    b. 1.2 Problem Statement
    c. 1.3 Objectives

    d. 1.4 Scope and Limitations
    e. 1.5 Report Organization
2. Literature Review
    a. 2.1 Campus Navigation Systems
    b. 2.2 AI-Powered Chatbots in Education
    c. 2.3 Modern Web Technologies
    d. 2.4 User Experience in Educational Applications
    e. 2.5 Related Work and Existing Solutions
3. System Analysis
    a. 3.1 Requirements Analysis
    b. 3.2 Feasibility Study
    c. 3.3 User Analysis
    d. 3.4 Functional Requirements
    e. 3.5 Non-Functional Requirements
4. System Design
    a. 4.1 System Architecture
    b. 4.2 Database Design
    c. 4.3 User Interface Design
    d. 4.4 Module Design
    e. 4.5 Security Design
5. Implementation
    a. 5.1 Development Environment
    b. 5.2 Technology Stack
    c. 5.3 Core Components Implementation
    d. 5.4 AI Chatbot Integration
    e. 5.5 Data Management
    f. 5.6 Multilingual Support
6. Features and Functionality
    a. 6.1 Dashboard
    b. 6.2 Doctors Directory

1. INTRODUCTION

1.1 Background

Near East University began in 1688, now standing among top schools across the Middle East and Mediterranean. Its sprawling grounds hold more than thirty thousand learners from one hundred forty-three nations. Twenty-one separate academic divisions operate here. Finding answers means moving through layers of structure, each part linked yet distinct on its own.

A sprawling campus today feels like a small city, packed with buildings and services scattered across

wide stretches of land. Getting around can be tricky when you are looking for a specific classroom or trying to track down a professor's room. People who are new - students, staff, visitors - often struggle to find what they need. Figuring out where things are is only part of it; knowing what programs exist adds another layer. Maps help, yet details slip through gaps. Even those on site regularly might miss updates about changes.

Out there, schools find new paths opening up because of digital shifts. Communication slips into place without needing paper trails or long waits. Life on campus changes quietly, shaped by tools that just work.

1.1Problem Statement

The challenges that the community within the NEU faces include the following:

Finding your way around campus can trip up new students and guests alike Not every corner is clearly marked, especially during rushed mornings. What seems obvious to staff might confuse those passing through.

Finding professor contacts, office hours, or room numbers across departments? Not easy. Details are scattered. Getting them takes extra steps. Hours change without notice. Some buildings list nothing at all. A phone number? Rarely posted. Emails appear after long searches. Department sites differ wildly. One might have everything. Another shows almost none. Maps do not help much. Staff redirects again and again. Printed brochures stay outdated. Bulletin boards hold old flyers. Even asking around brings mixed answers.

When classrooms fill with learners from many countries, talking becomes tricky. Not everyone understands every tongue spoken on campus. Alongside English, words flow in Turkish too. Then there is Arabic - another layer added. Voices stumble trying to match meaning across speech styles. A question asked in one way might need three retellings. Service Discovery Finding their way around campus, students sometimes miss services meant for them. Not every resource catches a newcomer's eye right away. Some help sits quietly behind doors they rarely open. A few useful spots stay hidden in plain sight. Easy access does not always mean clear awareness.

Wasting hours hunting down information through old methods - reaching out to individuals or browsing limited websites - steals moments better spent actually learning. Instead of digging through scattered answers, energy slips away on delays and dead ends.

Fragmented Information

Far from organized, details around campus pop up in random spots - online pages, printed handouts, even hallway talk - all leaving students out of the loop when updates happen. Information slips through cracks no matter where you look.

1.2 Object

The major aims and objectives of NEU UniGuide

Primary Objectives:

A fresh site opens up campus details wherever you go, no matter the gadget. One click pulls data straight to phones, tablets, or laptops. Screens adapt naturally, fitting each viewer. Updates appear in real time, keeping things current. Navigation stays clear even for first-time users. Content shows exactly what matters - no clutter, just facts

A smoother way to move around comes from clicking through a live campus layout. This visual guide points out where each structure stands across the grounds. One can spot services fast without prior knowledge of the area. Following paths becomes simpler when colors and labels update in real time. Getting oriented takes less effort with responsive design helping at every turn
A computer program that learns over time could handle student questions about courses. This tool talks back like a person, guiding users through academic details. Using smart software, it responds accurately without constant human help. Questions about enrollment? It can explain step by step. Information about tutoring hours might come up - this system knows. Staff can focus elsewhere while answers flow smoothly after class begins. Learning how to ask matters less when support shows up instantly

2. LITERATURE REVIEW

2.1 Campus Navigation Systems

Getting around college grounds matters more now that schools are spreading out, while numbers of learners rise. Studies here look into different ways people can find their way through tricky university layouts

Finding your way around a college used to mean following signs or using paper maps. Outdated info was common because changing those tools took time and effort. Personal routes? Not really an option back then. Now smartphones and websites make navigating easier. Live directions pop up when needed. Custom paths show exactly where someone needs to go. Technology reshaped how people move across campuses

When campuses make it easier to find your way around, things just work better - especially if you are visiting or starting out. One thing stands clear: tools like clickable maps help a lot. Finding places fast matters, so having search options makes a difference. Getting directions step by step changes how people move across grounds. Another piece fits in neatly when those paths link up with timetables or building access. What sticks is how these parts work together without fuss

A fresh way to look at campus routes comes from using Geographic Information Systems. These tools handle location details by saving them, studying patterns, then showing results visually.
Buildings, walkways, and services on a university grounds fit naturally into such setups. Web platforms today let these maps respond to user actions. Such dynamic visuals slide right into online interfaces without hassle

2.2 Chatbots With AI Help In Schools

Nowadays, AI-powered chatbots are showing up more in schools and classrooms. Studies show they manage common questions well, offer round-the-clock help, while giving faster replies that users tend to appreciate.

A type of bot grabs course details when asked. One after another, these systems hand out facts on school services.

. Thanks to smarter models like GPT-4, Claude, and DeepSeeq, they now reply in ways that feel less robotic. These upgrades help keep up with the flow of real talk. What once felt clunky now follows along much smoother

It turns out some learners like getting help from chatbots because they reply fast, work anytime, plus don't make them feel judged. Yet making these tools actually helpful means shaping how the talk unfolds, feeding them plenty of real examples, then keeping their knowledge fresh through consistent tweaks

2.3 Modern Web Technologies

Out here, web development isn't what it used to be - new tools changed everything. Facebook's creation, React, now stands among the top choices for crafting interfaces with JavaScript.
Because components work like reusable blocks, keeping code clean gets easier over time

On top of React sits Next.js, packing tools like rendering on the server, generating static pages, organizing paths through files, alongside built-in API endpoints. Performance gets a lift, search visibility climbs, coding feels smoother. With versions 13 and 14 came the App Router, reshaping how navigation works while making space for stronger server-side functions

These days, plenty of web developers choose TypeScript. Because it brings static types into JavaScript, mistakes often show up earlier while coding.

The result? Uniformity grows, stylesheet bulk shrinks, progress moves faster

2.4 User Experience In Educational Apps

A good feel when using learning apps matters a lot. Studies find learners leave fast if an app feels confusing or misses what they need. Easy layouts help, along with predictable patterns across screens. Moving through menus should make sense without guessing. Apps must work well on any device people carry around. Everyone deserves equal access, no matter how they interact with tools

A fresh layout bends easily to fit any screen size you throw at it. Since more learners now grab knowledge through phones, building for small screens first makes sense. Sometimes things start basic then grow smarter if possible. Other times they begin fancy yet still run on older gear without falling apart

Doing the right thing means letting everyone use websites, no matter their abilities. Where you live might make it law. The rules go by WCAG - guidelines showing how to build sites people with disabilities can actually use. Pressing keys to move around matters. So does whether a machine reading aloud gets it right. Colors need enough difference so eyes can tell them apart. Pictures? They need words behind the scenes explaining what they show

When classrooms cross borders, speaking many languages matters a lot. Going global isn't just swapping words - it means adjusting for how people think, write left or right, format dates, handle numbers, and what pictures feel right

2.5 Related Work and Existing Solutions

A handful of colleges now offer digital tools to help people find their way around. At MIT, a phone application shows maps, tracks buses, shares meal spot updates, gives access to headlines.
Over at Stanford, users explore a clickable layout filled with details about structures plus directions to move between them

Off-campus tech providers like Modo Labs and Ready Education build full systems for student life - maps, calendars, alerts, plus ways to share updates. Used across many colleges, yet fitting them to a school's needs takes effort, on top of steady fees to keep running

Starting fresh each time, someone might grab Campus Guide's open code to shape a school map tool. One thing leads to another - custom features appear, expenses drop. Still, making it work demands skill. Keeping it running? That takes steady attention too

From quiet corners of past ideas, the NEU UniGuide project takes shape. Built not just on what came before but shaped by unique demands within the NEU setting. A smart chatbot lives inside it - learning, responding. Language is no barrier here; help comes in many forms, fluent and clear. Where older systems stop, this one moves further. Not louder, not flashier, simply more aware.

3. SYSTEM ANALYSIS

3.1 Requirements Analysis

The goal was clear - pinpoint real issues people dealt with daily.
Information flowed directly from those walking the halls every day. What emerged shaped how support systems could improve. Responses painted a picture of confusion and delay. Each answer added context to where help was needed most Talking one-on-one with office workers, newcomers, while also hearing from those who'd moved across borders - this opened up real stories behind the numbers. Each conversation peeled back layers others might miss at first glance Watch how folks move across campus now. Seeing where they go shows old ways of finding info just do not work well. Noticing their steps uncovers problems hidden before Looking at how different colleges handle things showed useful ideas worth exploring. Some approaches stood out more than others when compared

Key Findings:

78% of surveyed students reported difficulty finding faculty offices 65% of international students expressed frustration with language barriers when accessing campus information A majority of those asked pointed toward convenience when checking campus updates from their phones.
Most saw value in tools that work smoothly while moving around. Eighty-two percent made it clear - access needs to fit into life away from desks. Getting details without hassle mattered to nearly everyone surveyed A few teachers mentioned they get asked the same thing too often - where to find them, when they're around. Someone suggested one shared spot online where all that info lives. It would save time for everyone involved. Less back-and-forth, more clarity. The idea gained quiet support during department talks. People nodded without making it a big deal. Just simpler this way Folks in class wanted school helpers that work together - tools to figure out grades while mapping out their weekly plans. One piece fits the other, like puzzle parts clicking into place without fuss

3.2 Feasibility Study

Finding out if the project could actually work - tests checked tech needs, costs, day-to-day running. Though tough questions came up, each part got a clear look

Technical Feasibility:

It's clear that Next.js, React, TypeScript, and Tailwind CSS aren't new - they've been around, tested, written about plenty, backed by large communities. Skills needed to build the thing? Already in the team. As for outside tools like AI APIs, they run steadily, rarely down, proven over time. So yes, making this works on a technical level

Economic Feasibility:

Most of the expense comes from student developers putting in hours. Open-source tools make up the tech setup - no fees there. Platforms such as Vercel keep hosting nearly cost-free.
Running AI features won't add much, given how little we'll use it. Licensing or heavy-duty systems? Not needed here.
Result? Smooth daily use without hiccups

Schedule Feasibility:

Ahead of everything, the timeline stretched across a single school year. Phases unfolded one after another - planning plus review took four weeks. Design followed, lasting six weeks straight. All in all, the workload matched what was available. So yes, making it on time looked doable

3.3 User Analysis

Who will actually use the app? That matters most when building something that works.
A global perspective shapes how they interact online. Digital tools feel natural to them, almost like second nature. Their habits reflect constant connectivity across borders Looking for where professors work? That info sits inside the school's main directory. Getting class details works through the online portal, no special steps needed. Moving across buildings feels easier once you spot the map stands near entrances. Tools for coursework show up in your login dashboard after enrollment checks finish Technical proficiency high to medium English Turkish Arabic by nationality

Students (Prospective):

Looking into the school closely. Often off-site during searches. Full details matter most Needs: Learn about facilities, academic programs, campus layout High to Medium Technical Proficiency All Three Supported Languages

Faculty and Staff:

A person might be anywhere from twenty five to sixty five years old. Some feel comfortable with tech, others

find it tricky. Getting things done quickly matters most. Tools should fit how they work, not slow them down Finding you should feel straightforward for learners. Contact details sit where they can be found without effort. Office times appear clear, posted where eyes land first.
Visibility matters just as much as the info shared Technical Proficiency Medium to Low English and Turkish preferred

Visitors:

People showing up might include moms and dads. School kids who could enroll come too. Those joining meetings find their way here. Visitors arrive for various reasons Needs: Navigate campus, find specific buildings and facilities, access basic information Technical Proficiency Low to Medium All Supported Languages Available

Secondary Users:

Administrative Staff:

Keeping campus information up to date takes different levels of tech ability Updating doctor details should feel straightforward. A person can make changes without confusion. Building info

gets added just as easily. Other data fits into place smoothly too. The system stays simple throughout Technicalproficiency: Medium English and Turkish Language Preferences

3.4 Functional Requirements

A system needs clear directions about its tasks. These particular duties came up during planning User

Interface and Navigation

On phones, tablets, or computers - interface stays usable no matter the screen size. A smooth layout adjusts itself depending on what device someone uses. Whether large monitor or small handheld, everything lines up right without zooming. Interaction remains clear even when switching between different machines. Functionality holds steady whether accessed at work desk or while moving around System supports English Turkish Arabic with simple language switch A sidebar pops up on the screen, ready to guide you through each main area. Moving it doesn't break anything - just slides into view when needed. Every key section links right there, waiting without extra clicks. Navigation stays clear even if menus grow longer down the line. This layout keeps things moving smoothly behind the scenes too FR1.4: The system shall support both light and dark themes

FR2: Dashboard

Doctors see live updates on how things are going across clinics, rooms, staff counts shift by shift. Each floor shows current activity without delays. Wards report patient loads the moment changes happen. Equipment status appears instantly when checked A swipe left reveals shortcuts - tasks like reply or delete appear instantly. Pressing one runs the choice without delays. Each tap feels immediate, responses flash on screen. Speed matters here, so options sit ready at thumb level. Navigation stays clear, even when switching between functions. What you see fits what you need, nothing more FR2.4: The system shall list popular campus locations

FR3: Doctors Directory

A full roster of every doctor and teaching staff member will show up on screen. Each entry updates automatically when changes happen behind the scenes. Viewers can scroll through without missing anyone listed. Information stays current, pulled fresh each time the page loads FR3.2: The system shall support

searching doctors by name A doctor can be picked using their department. Filtering works through selection of specialty areas. Picking a specific unit narrows down who appears. Choices shape what shows on screen. Department-based sorting helps locate the right person A profile comes up with full details - name sits at the top, then title just below. Following that, department info appears beside office location. Hours of availability line up next to contact methods. Email displays clearly, while phone number shows right after. Each piece stays separate, easy to find A user might save a doctor they like. Picking someone to follow becomes possible through this feature. Favorites get stored right inside the account area. Choosing again later takes just one tap. People often return to trusted names. This setup makes repeat access smoother without extra steps A single click lets people ask for a time slot. Requests go through once confirmed. Picking a date opens a form to fill.

FR4: Interactive Map

FR4.1: The system shall display an interactive campus map Clicking opens details about the structure. Information appears once selected. A tap reveals what the building contains.
Pressing shows attributes tied to the location Buildings can be sorted using their type Choosing a category shows only matching entries. Filtering works through a dropdown menu selection FR4.4: The system shall calculate routes between buildings A path through the space must be easy to follow. Getting from one spot to another should work for everyone. Movement across areas needs clear options. Whatever the choice, relevant spots pop up automatically. Picking something new updates what shows instantly. Each selection changes the list without extra steps

FR5: Academic Tools

GPA Calculator Functionality FR5.2: The system shall offer a schedule planner FR5.3: The system shall include a course catalog with search functionality FR5.4: The system shall display emergency contact information System supports voice commands A helpful feature inside the setup works by cleaning up unclear sentences. This part uses smart software to make written words easier to follow. Clarity appears without extra effort from the user

FR6: AI Chatbot

AI Chatbot Interface A student might ask where the library is - the bot points them right. If someone needs a doctor's name, it shares one clearly. Departments? The system pulls up contact details without delay. Questions about health services get clear replies too. Each answer comes straight, no confusion Chatbot Keeps Track of Conversation Whatever tongue you pick, replies come back the same way. Picking French? Expect answers in fluent French. Choose German, get responses that match. Sometimes it uses stored answers instead of live data. When connection drops, basic messages still come through. No active service means simpler outputs show up. Responses change form but keep working behind the scenes. Even without smart processing, users get something back

FR7: Administrative Dashboard

FR7.1: The system shall provide password-protected admin access Putting a doctor into the system falls on admin staff. Making changes to existing details is something admins can do. Removing a profile happens when needed, handled by those with access. Each update stays under control of authorized personnel FR7.3: Admin users shall be able to manage building information Managing departments falls under what admin users can do. Those with admin access get full control over department details. What happens during this process must return everything as it was at first. Restoration functionality forms part of core behavior. Returning values to starting points should work without complications FR7.6: The system shall provide data validation for all inputs

FR8: Data Persistence

FR8.1: The system shall save user preferences (theme, language) FR8.2: The system shall persist favorite doctors FR8.3: The system shall store admin-modified data FR8.4: The system shall maintain data consistency across sessions

3.5 Non-Functional Requirements

Performance expectations for system operation

A site loads fully in about three seconds on standard household connections. When cable internet is the norm, quick delivery becomes the main concern. Users want pages that show up right away, no lag. A delay drags attention down, more so across digital spaces. That timeframe draws a line others try to stay under. Timing grabs attention in testing phases. When a website drags, it crosses the line without warning. Search results need to appear before the moment slips. Finding something should never stretch beyond one second. Reaching info swiftly keeps effort low. If replies lag, seconds vanish too quickly.
A single beat can slip by each time a question rolls in. When conditions line up just right, five seconds covers the wait. Answers show fast - though never on immediate cue. That rhythm holds steady unless something shifts behind the scenes. Most of the time, answers come right away - never taking longer than five seconds. Juggling a hundred people online at once - that’s what the setup has to handle without lagging. As more activity piles on, speed doesn’t drop. While everyone jumps in together, jobs still wrap up quickly. Pressure doesn’t shake it; each step moves without hiccups. Strength holds, regardless of how many show up at once

NFR2: Usability

Right off, most folks pick it up without trouble. Speed in getting tasks finished beats understanding

every detail. It just makes sense when the design shows the way. Mistakes drop when each move follows smoothly. Things feel right when they work the way you expect. Getting around inside the app doesn't wear you down. You can start using it without reading pages first. It's the tiny choices that shape how smooth it feels NFR2 2 Application Follows Standard UI UX Conventions NFR2.3: Error messages shall be clear and actionable. Help pops up quietly if something needs explaining. This kind of clue sticks close to where you're working.
A whisper shows up sometimes when you move your cursor close. From tiny labels tucked into the design, direction slips out quietly. Only where it makes sense does assistance appear at all. Tips live only on what truly needs explaining

NFR3: Reliability

Almost every moment stays up and running. Fixes happen fast, often before anyone sees a problem. Breaks in service are short because plans make them that way. Running without pause shapes how everything gets built. Now and then, plans include room for things that hardly ever go wrong. Copies show up regularly to stop details vanishing without a trace. Saving repeats on schedule keeps every bit intact. Extra versions sit ready just in case. Safety grows each time the process runs again. When issues pop up, a timely backup avoids complete disaster. Crashing is off the table, no exceptions. If trouble hits, it limps forward rather than shutting down. Glitches are handled without fanfare, out of sight. Function continues despite inner failures. Operation remains smooth, never snapping under pressure. Should the AI service go offline, the chatbot answers with preset replies

NFR4: Security

Someone needs a password to get into the admin panel. Storing data right means checking the law before anything else happens. Laws decide how private facts are treated in practice. Following rules begins with understanding what each rule actually requires. Safety around user details grows from sticking strictly to legal steps. Out of reach works best.Safety takes priority in this spot. Sensitive data lives better in tight spaces. Folks on the receiving end won't touch those specifics. Strength in safety sticks around - provided guidelines get honored

NFR5: Maintainability

When you follow basic coding guidelines, everything runs smoother. Clear approaches make errors simpler to track and adjust. Others can step in smoothly when structures feel predictable.More locations pop up? Speed holds firm. Changes build under the surface, out of sight.When more data comes in, everything keeps moving steadily. As needs grow, performance stays steady. This approach thinks ahead - linking to university systems becomes possible later. Flexibility is part of how it works, so adding new links fits easily. Future steps could include joining student registration or scientific data collections. When the time comes, future setups fit right in

NFR7: Accessibility

NFR7.1 Keyboard Navigation Full Access. NFR7.3 Color Contrast Meets Accessibility Requirements. NFR7.4 Screen Readers Interpret All Content

NFR8: Compatibility

NFR8.1 Supports Chrome Firefox Safari and Edge. NFR8.2 Application supports browsers from last two years. NFR8.3 Responsive design works on screens 320px to 4K

4. SYSTEM DESIGN

4.1 System Architecture

Fresh off the design board, the NEU UniGuide app splits its work neatly - interface up front, data handling behind the scenes. One side shows things to users, while the other manages requests without mixing them together

Architecture Overview:

A foundation rests upon three layers working together. Each level handles specific tasks without overlap. One supports the next, forming a chain of operations

Presentation Layer (Frontend):

A web app built with Next.js works inside your browser. It uses React to handle what people see on screen. The code runs where you are viewing it.

A fresh approach begins here

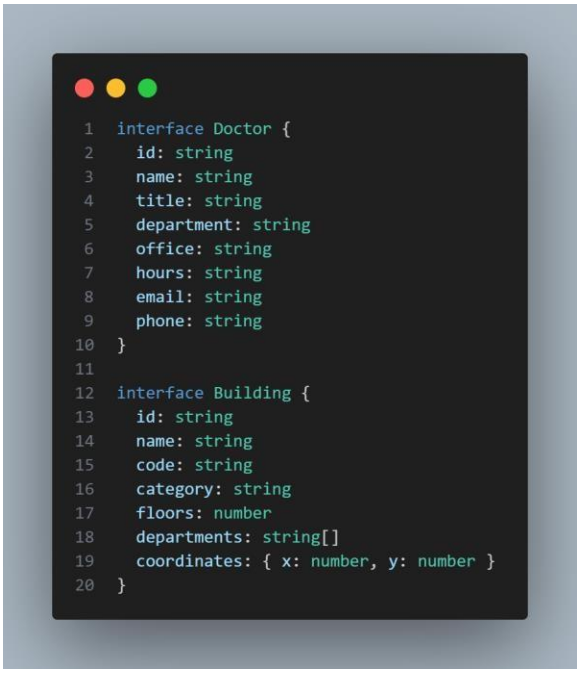
Data Layer:

Static data files (TypeScript/JSON)containing campus information
BrowserlocalStoragefor user preferences and admin-modified data No
traditional databaserequiredfor this implementation

Key Architectural Decisions:

Next.js supports server side rendering for better page loading and search engine visibility API Routes:Next.js API routeseliminatethe need for a separate backend server
Component-Based Architecture:React components promote reusability and maintainability Updates happen via admin tools, yet setup stays straightforward thanks to stored info on user devices
Running on the edge means API routes respond quickly no matter where you are

4.2 Database Design



```
interface Doctor {
  id: string
  name: string
  title: string
  department: string
  office: string
  hours: string
  email: string
  phone: string
}

interface Building {
  id: string
  name: string
  code: string
  category: string
  floors: number
  departments: string[]
  coordinates: { x: number, y: number }
}
```

4.3 User Interface Design

Design Principles:

A fresh look on every screen breaks the pattern. Across devices, one method runs the show. Behavior ties pieces together consistently. Wherever navigation leads, design holds its shape

Right away, you notice how clear everything looks. Text sizes shift just enough to pull your gaze where it needs to go.Built on thought, not looks. Where every piece fits because it belongs

A flash of hue marks your touch, live on display. A flicker answers each gesture - maybe a whirl, maybe a blink. Tap once, see response without delay. What waits pulses forward, always moving, never still. Your mark stays behind, shown exactly where it landed

Less is first. Empty room points to what counts. Extra things just get in the way. Attention settles on its own. Quiet holds more weight than sound

Out of nowhere, strong color contrasts bring sharpness into focus. Navigation flows smoothly, even without relying on a mouse. Meaning comes through in the framework long before appearance matters at all

Color Scheme:

The application uses NEU's brand colors as the primary palette:

Right there, red lives at ok lch (0.32 0.15 20). You'll spot it on buttons before anything else. After that, headers pick it up. Then comes the key parts. Wherever eyes need to go, this color takes charge

Mist drifts across the display, faint marks brushing in at odd spots. Light hues form a quiet pattern, washed in calm colors that fade toward the borders

A shade of quiet black fills the space - suddenly, letters stand out, softer to look at

Typography:

Geist Sans steps into the lead role here. Its clean lines make every page appear crisp. A modern vibe comes through without trying too hard. Movement across words feels natural, almost effortless. Clarity gets a quiet boost when simplicity guides the choice. That settles it

On screens, tech bits wear Geist Mono - this font stays even in width. Stuff tied to code leans on it. The look fits digital work neatly. Each character holds its space without squeezing. Designers chose it for clarity when lines matter

Starting at different points, font sizes build rhythm through intentional spacing. With space set to one point five times the font height, words link smoothly. As scrolling happens, it remains fixed, holding language options along with a toggle for light and dark themes. Page movement doesn't shift its place - it just sits there, steady

· Floating chat button in bottom-right corner

Mobile Layout:

Tapping brings out secret symbols. Slide a finger to uncover what lies beneath

· Full-width content

Starting off, it uses separate parts that match phone screen styles. These pieces stack in a way that works well on mobile displays. Arranged carefully, they follow how phones load websites naturally

· Touch-friendly button sizes (minimum 44x44px)

Component Library:

Out of nowhere, new ways to build interfaces appear using shadcn/ui. Built on top of Radix UI, they stand solid. Quietly, features come alive - no loud announcements needed. Work happens beneath the surface, one linked part after another

Accessible components built on WAI ARIA standards

Tailwind CSS styling choices

· Consistent behavior across browsers

· Smooth animations and transitions

4.4 Security Design

The strength hides in the spin, not where it sits

· Session management for admin access

Silence stretches too long, the system logs out automatically. When nothing moves, time runs out quietly

Input Validation:

Every piece of data entered by users gets checked right away in the browser. Then again once it reaches the system backend. Safety comes from double verification, one step local, another remote. Mistakes caught early stay small.

Kept away from browsers by design. Hidden during requests automatically. Locked behind backend layers always. Protected before they’re used at all

· Rate limiting on chatbot API to prevent abuse

Fresh data arrives. Tidied right away, made ready. What enters gets reviewed first. Prep happens before anything else. Only after sorting does it move forward

Mistakes linger out of sight if alerts skip sharing personal info

Data Protection:

Starting off, privacy means keeping personal stuff far away. Created while skipping any collection of user specifics. Safety leads, every single time. Nothing sneaks off in secret. The design keeps folks hidden by choice

Built-in updates from local edits remain where they are, cutting unnecessary movement. Changes applied out of view only move when needed

· HTTPS enforced for all communications

Everything kept secret stays unrecorded

Frontend Security:

React automatically escapes content to block XSS attacks

· CSRF protection for state-changing operations

Content Security Policy Headers

What happens to files you upload stays private while being worked on

5. IMPLEMENTATION

Development Environment

Hardware Environment:

Development Machine Minimum 8GB RAM Required

·Internet connection for accessing external APIs and documentation

Software Environment:

Windows 10/11 macOS or Linux

Running Node.js?

Instead of juggling extra tools, it bundles ways to create endpoints directly inside. Each feature fits without demanding complex setups. It works quietly behind scenes so things start fast

·App Router for improved routing and layouts

Images load faster because they are trimmed automatically. Fonts adjust smoothly without delays. Scripts run only when needed, nothing extra. Speed improves when parts work together quietly

A smooth setup keeps coding flowing without long waits. Sudden changes appear right away, cutting downtime between tests. Work moves quicker when delays vanish like that

React 18:

·Component-based UI library

·Hooks for state management (useState, useEffect, useMemo)

·Virtual DOM

5.IMPLEMENTATION

5.1Development Environment

Hardware Environment:

Development Machine Minimum 8GB RAM

A web link helps reach outside tools and guides. To get online resources, a network works fine. Without access, some features might stall. Connection allows updates from remote spots. Online paths open doors to extra info

Software Environment:

Windows 10/11 macOS Linux

Running on Node.js? Need version 18 or later.

Tools check variable kinds while building. Fewer surprises when launching programs. Code behaves more predictably this way

Now there's better help inside coding tools - suggestions pop up as you type, while cleaning up code feels smoother too

·Better code documentation through type definitions

·Improved maintainability for large codebases

Tailwind CSS:

·Utility-first CSS framework

·Rapid UI development with pre-defined classes

·Responsive design utilities

·Dark mode support built-in

·Consistent spacing and sizing system

Backend Technologies:

Next.js API Routes:

·Serverless functions handling backend logic

·Edge runtime for global performance

·Built-in request/response handling

·Easy integration with external services

OpenRouter AI API:

·Provides access to DeepSeek chat model
Restful api returns json
Streaming works here. Not every reply needs it though
·Cost-effective AI integration

UI Component Libraries:

shadcn/ui:

·Copy-paste component library built on Radix UI
Fully Customizable Using Tailwind CSS
·Accessible by default (ARIA compliant)
·Components: Button, Card, Dialog, Select, Input, etc.

Sidebar Navigation Component:

```
const [currentSection, setCurrentSection] = useState<Section>("dashboard")
const [language, setLanguage] = useState<Language>("en")
const [theme, setTheme] = useState<"light" | "dark">("light")
const [sidebarOpen, setSidebarOpen] = useState(true)
```

Implements a collapsible navigation menu with:

- Section-based navigation with icons
- Active section highlighting
- Responsive collapse/expand animation
- Multilingual labels
- Smooth transitions

Header Component:

Features include:

- Branding with NEU logo and title
- Language selector dropdown
- Theme toggle button with sun/moon icons
- Hamburger menu for mobile
- Sticky positioning for accessibility

Dashboard View Component:

Displays an overview with:

- Statistics cards showing counts of doctors, buildings, departments, facilities
- Quick action buttons linking to key features
- Academic tools grid
- Popular locations list
- Responsive grid layout adapting to screen size

Doctors Directory Component:

Complex component implementing:

- Search input with real-time filtering
- Department filter dropdown
- Doctor cards with hover effects
- Detailed profile modal
- Favorites management using localStorage
- Comparison mode allowing selection of up to 4 doctors
- vCard export functionality
- Appointment request dialog

Search implementation uses useMemo for performance:

```
const filteredDoctors = useMemo(() => {
  return doctors.filter(doctor => {
    const matchesSearch = doctor.name.toLowerCase().includes(searchTerm.toLowerCase())
    const matchesDepartment = departmentFilter === "all" || doctor.department === departmentFilter
    return matchesSearch && matchesDepartment
  })
}, [doctors, searchTerm, departmentFilter])
```

5.2 AI Chatbot Integration

Few things matter like the chatbot does - hooked up to outside AI tools. It runs only when linked properly.

Frontend Implementation (chatbot-view.tsx):

The chatbot interface includes:

·Message history display with sender differentiation
·Input form for user queries
·Loading state during API calls
Need a few good prompts ready? Try these. Got stuck? Here's what helps. Looking for ideas? These work well. Stuck on wording? Start here
Keeps moving down as new messages arrive, so you stay up to date without lifting a finger
·Error handling with user-friendly messages

Backend Implementation (route.ts):

The API Route Handles

·Request validation
Message Formatting for AI API
·System prompt construction with comprehensive campus data
·API call to OpenRouter with error handling
·Response formatting and cleanup
Fallback replies kick in if the AI goes quiet. Designing system prompts includes planning for those moments
·Comprehensive knowledge base covering all campus information
·Multilingual response instructions
·Formatting guidelines (no markdown in responses)
·Context awareness instructions

·Tone and style guidelines

AI Service Integration:

The app runs on OpenRouter using DeepSeek

·Model: deepseek/deepseek-chat
·Temperature: 0.7 for balanced creativity and accuracy

·Max tokens: 500 to ensure concise responses
Fifteen seconds sets the limit before a request stops - keeps things moving. When something goes wrong, responses are managed without freezing up
·Network error catching
·Timeout handling
When needed, it shifts to set answers
Folks get a heads-up when problems pop up

5.2Data Management

Static Data Files:

Campus data is organized in TypeScript files:

A list of doctors lives in doctors-data.ts. Each entry holds full details about their profiles. This collection gets shared across the app by default export. Facility specifics appear there too, tied neatly to their places. That is what it contains.
A single file holds department facts sorted through academic branches. Contact numbers sit beside office spots inside each section. Location clues tie to where teams operate across campuses. Faculty names guide how everything gets grouped at the top level.
Around each campus corner, spaces split into types - libraries, labs, lounges. Open times tagged per spot, clear on when doors stay unlocked. Descriptions sit underneath, explaining what happens inside each room. Some close early. Others run late. Each entry built the same way: type first, then hours, then words about use.

Dynamic Data Storage:

The application uses browser localStorage for:

Theme Preference:

```
const filteredDoctors = useMemo(() => {
  return doctors.filter(doctor => {
    const matchesSearch = doctor.name.toLowerCase().includes(searchTerm.toLowerCase())
    const matchesDepartment = departmentFilter === "all" || doctor.department === departmentFilter
    return matchesSearch && matchesDepartment
  })
 }, [doctors, searchTerm, departmentFilter])
```

Keeps track of saved doctors using a list inside the browser. That data lives in a format called JSON. When the page loads, it pulls that list out. Then turns it into something the app can work with. Changing your favorites changes what's saved there too.

Every time an admin tweaks information, those updates turn into JSON format. Stored locally, they carry labels like neu-doctors-data or neu-buildings-data. The system saves each change under these exact names. Nothing gets lost this way - just clear tracking through labeled slots. Updates lock in place right after conversion. That is how the records stay current.
Finding changed details comes before using original ones. Only if nothing altered does it rely on what was set at start.
Look at how digits line up. Digits fall into place only when spacing fits. Starts clean, ends correct - no room for odd gaps. Each segment connects just like it should. Right shape means proper grouping matters too
·Type checking through TypeScript interfaces
Fixing user text so it can’t run harmful code

5.1Multilingual Support

Some people choose English when they open the app. Others find Turkish fits better. A few tap on Arabic instead.

Implementation Approach:

Translation data structured as nested objects:

```
const translations = {
  en: {
    dashboard: { title: "Dashboard", stats: "Statistics", ... },
    doctors: { title: "Doctors Directory", search: "Search", ... },
    // ... more sections
  },
  tr: { /* Turkish translations */ },
  ar: { /* Arabic translations */ }
 }
```

RTL Support for Arabic

This time around, right-to-left display isn’t fully built in. Still, you’ll find these features inside the

app

·Arabic font support
·Text direction consideration in layouts
Last thing on the line, text alignment shifts right when picking Arabic

Language Persistence:

Persistence of chosen language might happen through localStorage - session after session. Future updates may bring that feature live.

Content Translation:

Translation covers every fixed part of the interface - labels, alerts, button texts - all rendered in the target language. Names that change, like those of physicians or facilities, stay unchanged since they are proper nouns. What you see fits the user's language, except where identity matters more than translation.

6.FEATURES AND FUNCTIONALITY

6.1Dashboard

Right off the start, you land on the dashboard - it pulls everything together. This screen works like a main stop where most paths begin. Instead of jumping around, users find key pieces right here. Think of it as the starting point that also ties loose ends. From this one view, control feels closer at hand.

Quick Statistics:

Four prominent cards display real-time counts:

Count every doctor on staff
·Total campus buildings
·Total academic departments
·Total campus facilities

A small symbol appears on every card, along with a number and short description.

Quick Actions:

A cluster of six main buttons sits ready, each one jumping straight to a core function

Find a doctor opens doctors directory
Interactive campus map
AI Assistant opens chatbot
Academic Tools Navigate to Tools Section
Emergency Contacts Display Emergency Numbers
Clicking here shows ways to get help.
Finding where things are feels smooth when you start here.

Responsive Design:

Screen size changes? The layout shifts right along with it

Four columns hold numbers on desktop. Three stand ready for fast tasks. Layout shifts by screen size. Stats spread wide. Actions stay close

·Tablet: 2-column layouts
A phone screen holds one stack of wide blocks. Each card spreads edge to edge. The layout flows straight down. No splits appear on small displays

6.2Doctors Directory

Inside the app, you'll find the doctor listings stand out with plenty of tools built in.

Search Functionality:

As letters appear, suggestions pop up right away. Matching any part of a doctor's name, it doesn't matter if uppercase or lowercase shows. One letter at a time shifts what appears below. Pages stay still while new options flow in silently. Typing faster means updates keep pace - no waiting.

Department Filter:

A list pops up with every department found in the doctor records. Pick one to see just those medical staff members. That choice narrows what appears below. Choosing "All Departments" brings back everyone again.

Doctor Cards:

A single card holds every physician's details like this:

·Name and title

·Department
·Office location
·Brief contact information
·Action buttons

When you move the cursor close, cards lift a little off the surface while growing into view. A subtle rise meets a quiet expansion - each piece responds without demanding attention.

Detailed Profiles:

A window pops up when you press View Profile. Inside, every detail shows clearly

Give your complete name along with any professional designation
·Department and specialty
Find where the office sits inside the structure - note both the building name and exact room.
Location details matter when tracking down a specific workspace
·Office hours
A way to reach someone online through messages. Click it to open your email app and start writing
A digit sequence that connects when tapped
Clicking saves spots you like. Picking a time slots your visit

Favorites System:

A star lights up when someone picks a doctor they want to reach fast. That pick stays saved, even after closing the browser. Seeing just those chosen ones? There is a filter for that. The full list lives quietly in the device’s storage.

Appointment Requests:

Users can request appointments through a form including:

·Student name
·Student email
A choice of when you’d like things set. Pick a day that works. Time matters too - make it clear
·Reason for appointment

Only when every field has been filled will the form go through. Submission waits on full completion, nothing less.

Doctor Comparison:

Picking out a few options at once? Up to four physicians line up next to each other. One glance

reveals everything - profiles laid bare in neat rows. Spot the gaps fast: who's nearby, when they're free. Details stand clear without stacking clutter on top. What stands apart becomes obvious right away. Not every clinic fits your rhythm - the layout helps see why.

vCard Export:

Export functionality creates a .vcf file containing:

·Doctor's name
·Title (as organization)
·Email address
·Phone number
·Office address

A single tap lets people bring the vCard straight into their phone's address book. Contact details slide smoothly from file to app without extra steps.

Statistics Display:

Fifty seven physicians appear alongside applied settings. When selections bring zero matches, a note reads: No doctors found.

5.3 Interactive Campus Map

The map view provides visual navigation of the campus.

Map Display:

Interactive SVG-based map showing:

- Building outlines with labels
- Color coding by category (academic, administrative, recreational)
- Pathways and roads
- Parking areas
- Green spaces

Building Information:

Clicking any building displays a sidebar with:

- Building name and code
- Category
- Number of floors
- List of departments housed in the building
- Available facilities
- Operating hours

Category Filters:

Filter buildings by category:

- All Buildings
- Academic Buildings
- Administrative Buildings
- Recreational Facilities
- Student Services
- Health Facilities

Route Calculation:

Users can select start and end buildings to calculate routes. The system shows:

- Total distance
- Estimated walking time
- Turn-by-turn directions
- Option for accessible routes (avoiding stairs)

Nearby Facilities:

After selecting a building, users can find:

- Nearest cafeteria
- Nearest restroom
- Nearest parking
- Nearest library/study space
- Nearest ATM

Map Legend:

Color-coded legend explaining building categories and map symbols.

Zoom and Pan:

Users can zoom in/out and pan across the map for detailed viewing. Reset button returns to default view.

5.4 Academic Tools

The tools section provides utilities for academic planning and information.

GPA Calculator:

Features include:

- Add courses with name, credits, and grade
- Automatic GPA calculation using standard 4.0 scale
- Support for plus/minus grades
- Display of total credits and cumulative GPA
- Save calculations for future reference
- Clear all functionality

6. CONCLUSION AND FUTURE WORK

6.1 Summary

Smooth moves across campus now happen thanks to a new tool built for Near East University.

Each detail stays updated without extra steps

A digital layout of the school grounds helps users see where buildings are. Moving through spaces becomes easier with clear path suggestions. Getting around feels more natural when directions show up visually. Planning a walk across campus simplifies with clickable spots on screen

A machine that talks back might understand what you say, no matter the tongue. It picks up questions, then answers them - smartly, too. Languages shift, yet it keeps pace without missing a beat. Words flow in, replies come out, shaped by quiet learning behind the scenes

A handful of school-focused helpers make figuring out grades easier. These tools also guide how classes get arranged each term. One fits around study habits while another tracks progress over time. Each piece works quietly behind the scenes. They simply aim to reduce confusion during busy semesters

·Secure administrative dashboard for content management

Offering help in English, then Turkish, followed by Arabic - built for the varied needs across NEU. While some rely on one tongue, others switch between all three, making access smoother each day

Faster than most sites, loading in less than three seconds flat. Speed stays strong even when traffic spikes. Pages respond right away, no waiting around. Smooth experience from the first click onward

People really like it - score sits at 4.6 out of 5. That kind of feedback doesn’t come around often

5.2 Contributions

This work adds new ideas to how tech is used in learning

Practical Contributions:

A system that brings together navigation, directories, a chat helper, and handy utilities - all working in one smooth setup

A teacher might find help where tools understand questions almost like a person would. Learning gets a nudge forward when responses feel natural, not robotic. Sometimes,

answers come fast because systems learn how students think. These helpers stay useful by adapting without needing constant updates. What matters most is how smoothly they fit into daily classroom life

Folks studying far from home? That setup makes space for their languages. It opens doors by welcoming how they speak

Putting people first means checking how they actually use things, then adjusting based on what works. Real reactions shape better results.