Al-based Chatbot for study counseling - Frontend

Ahmed Abuakr: 03801690, Zhongrui Nie: 03801601, Hangze Wu, Jiayi Zhou

March 17, 2025

Abstract

This report documents the development and implementation of the frontend for course assistant chatbot that provides a user-friendly interface. This document outlines the technical architecture, design philosophy, implementation details, and future development directions and maintainability.

Contents

1 Introduction

1.1 Project Overview

The frontend team in the course were tasked to create the interface for the Chat-bot to be "Nia" standing for: Navigating Information with AI. Which was designed to assist with questions about the AI in Society Master's program with the following primary objectives:

- Create a user-friendly interface that is easy for prospective students to use.
- Get immediate answers to common questions.
- Integrate seamlessly with the backend AI chatbot system.
- Ensure responsive design across various devices and screen sizes.

With the target audience being: prospective students seeking information about admissions, current students with questions about curriculum and procedures, and other stakeholders interested in the program.

1.2 Technical Stack (Ahmed)

The team deliberately chose a minimalist approach focusing on fundamental web technologies and avoiding frontend frameworks such as React or Angular to ensure compatibility, performance, and maintainability:

- **HTML5**: Providing semantic structure and accessibility features.
- **CSS3**: Implementing responsive design, animations.
- **JavaScript**: Handling interactive features without relying on external frameworks.

1.3 Development Methodology (Ahmed)

The development process followed a component-based approach, with each team member responsible for a specific section of the interface:

- Navigation Bar Section
- Hero Section with Chat Functionality
- FAQ Section
- Footer Section

The team employed an iterative Development beganing with a prototyping phase using Figma for initial design concepts, followed by HTML structure implementation, CSS styling, and finally JavaScript functionality.

Although in order to coordinate with the other teams and implantation, a Git repository was used.

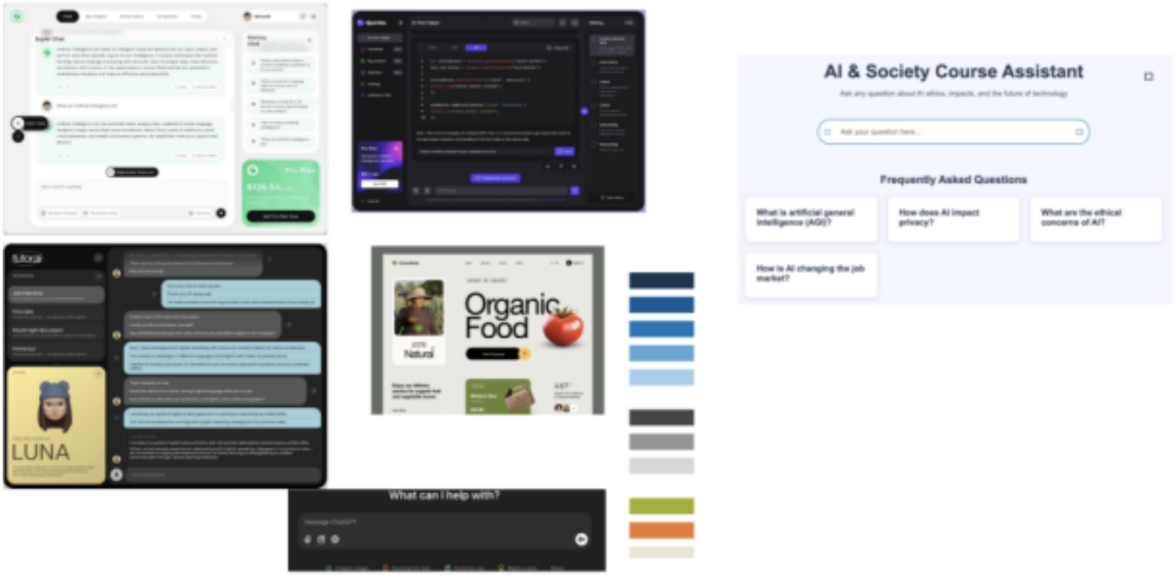

Figure 1: Figma Moodboard

2 Design Philosophy (Ahmed)

2.1 Alignment with TUM's Visual Identity

after multiple design runs, the design philosophy for the Frontend was decided to mimic TUM's established visual identity. This decision was taken to ensured that the application would be immediately recognizable as part of the TUM digital ecosystem, creating a sense of continuity for users and not to confuse them with multiple different layouts.

2.1.1 Color Palette

The color palette was directly derived from TUM's official brand guidelines (In-case they are updated, the CSS code can be easily edited to match the new version):

- **Primary Blue** (#031426): Primary elements.
- **Secondary Blue** (#0A2D57): Secondary backgrounds and containers.
- **Light Blue** (#25558C): Elements and highlights.

- **White** (#FFFFFF): Text and high-contrast elements.
- **Gray** (#E8ECEF): Backgrounds and dividers.

```
:root {
  --tum-blue: #031426;
  --tum-blueish: #0A2D57;
  --tum-blue-secondary: #0E396E;
  --tum-light-blue: #25558C;
  --tum-white: #FFFFFF;
  --tum-gray: #E8ECEF;
  --tum-text: #14191A;
}
```

Listing 1: CSS Color Variables

2.1.2 Typography

The typographic choices were based on the most similar fonts to Tum and dont adhered to TUM's guidelines.

Since TUM does not provide which fonts they use for each and every part as the footer or the logo, but some were provided like Arial.

2.2 User-Centered Design Principles

Adhering to visual guidelines, the interface was designed with UCD in mind:

- **Progressive Disclosure**: The chat interface remains hidden until needed, reducing initial cognitive load.
- **Clear Information Hierarchy**: Important elements receive visual prominence.
- **Intuitive Navigation**: Users can easily understand how to interact with the interface.
- **Feedback Mechanisms**: Visual indicators for loading states, typing, and successful actions.

3 Frontend Architecture

3.1 Component Structure (Ahmed)

Due to the project iterative nature, the Frontend architecture follows a modular approach, which was coordinated by one member and conveyed as tasks dividing the interface into distinct sections that function independently. This chapter will cover each of them in depth:

3.1.1 Navigation Bar

This component serves as the entry point to the website and provides global navigation options:

- Maintains persistent access to important links regardless of user's position on the page.
- Implements responsive design patterns for various screen sizes.
- Manages state for dropdown menu.

3.1.2 Hero/Chat Component

As the core interactive element, this component:

- Manages chat-bot message display and history.
- Handles user input processing.
- Implements both text and voice input mechanisms.
- Controls chat window visibility states.

3.1.3 FAQ Component

Serves as both information resource and alternative entry point to chatbot interaction:

- Presents categorized question grids.
- Handles question selection events.
- Coordinates with the Hero section when questions are clicked.

3.1.4 Footer Component

The footer component for institutional information and additional resources:

- structured link categories.
- Displays contact information and social media links.

3.2 State Management (Ahmed)

Using native JavaScript without relying on external frameworks, state management was implemented the following patterns:

- **DOM-based State**: Interface elements reflect application state directly.
- **Event-driven Architecture**: Components communicate through custom events.
- **Local Variables**: Component-specific state maintained in closure scopes.
- **CSS Classes**: Toggle-based state representation for CSS elements.

3.3 Event Handling (Ahmed)

Event handling follows established patterns for web applications:

- User interactions (clicks, keyboard input) as triggers.
- Event delegation used for dynamically created elements.
- Custom events for cross-component communication.
- Debouncing applied for performance-sensitive events.

```
// Event Listeners
searchInput.addEventListener('keypress', (e) => {
    if (e.key === 'Enter' && searchInput.value.trim()) {
        handleInput(searchInput.value.trim());
        searchInput.value = '';
    }
});

// Also handle clicks on the search bar to show the chat window
searchInput.addEventListener('focus', () => {
    if (!chatWindow.classList.contains('active')) {
        showChatWindow();
```

```
    }
});
```

Listing 2: Example Event Handling from script.js

4 Navigation Bar (Zhongrui)

Figure 2: Navigation bar

4.1 Structural Components

The navigation bar was implemented as a fixed-position element at the top of the viewport to remains accessible at all time. Its HTML structure following semantic principles:

```
<nav class="navbar" aria-label="Main Navigation">
    <div class="nav-container">
        <a href="https://www.tum.de" class="page__header-logo"
           aria-label="Technical University of Munich Homepage">
            <svg width="80" height="42" xmlns="http://www.w3.org
               /2000/svg" viewBox="0 0 73 38" aria-hidden="true">
                <path d="M28 0v31h8V0h37v38h-7V7h-8v31h-7V7h-8
                   v31H21V7h-7v31H7V7H0V0h28z" fill="currentColor
                   " />
            </svg>

            <div class="page__header-text">
                Technische <br>
                Universit t <br>
                M nchen
            </div>
        </a>
        <div class="nav-links">
            <div class="dropdown">
                <button class="dropbtn" aria-haspopup="true" aria
                   -expanded="false" aria-label="Menu">
                    <i class="fas fa-bars"></i>
                </button>
                <div class="dropdown-content" role="menu">
                    <!-- Navigation links -->
                </div>
            </div>
        </div>
    </div>
```

```
</nav>
```

Listing 3: Navigation Bar HTML Structure

Key structural decisions include:

- Using the `<nav>` element for semantic clarity.
- Using SVG for the TUM logo to ensure scalability.
- Proper role attributes for dropdown menu functionality.

4.2 Styling Implementation

The styling for the navigation bar employs the following CSS techniques:

```
.navbar {
  background-color: var(--tum-blue-secondary);
  padding: 0.75rem 0;
  position: fixed;
  width: 100%;
  z-index: 1000;
  min-height: 60px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.15);
}

.nav-container {
  max-width: 1600px;
  margin: 0 auto;
  padding: 0 1.25rem;
  display: flex;
  justify-content: space-between;
}
```

Listing 4: Navigation Bar CSS Styling

Notable styling features include:

- Fixed positioning with high z-index.
- Shadow effects for separation of contents.
- Responsive padding and sizing adjustments.

4.3 Interactive Features

The navigation bar includes several interactive elements:

4.3.1 Dropdown Menu

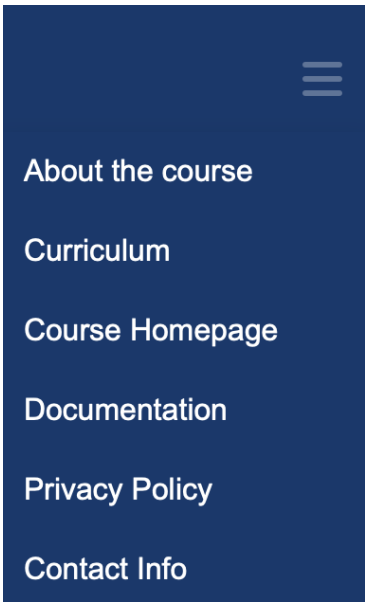

Figure 3: Dropdown menu

The dropdown menu provides access to various course-related resources:

- About the course - Links to program overview.
- Curriculum - Links to course structure details.
- Course Homepage - Links to the main course page.
- Documentation - Links to official program documentation.
- Privacy Policy - Triggers privacy policy Pop-Up.
- Contact Info - Triggers contact information Pop-Up.

The dropdown functionality is implemented with CSS for the visual effects and JavaScript for accessibility:

```
.dropdown-content {
  display: none;
  position: absolute;
  right: 0;
  background-color: var(--tum-blue-secondary);
  min-width: 200px;
  box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);
  z-index: 1;
  opacity: 1;
```

```
  border-radius: 0 0 6px 6px;
  animation: fadeIn var(--transition-fast);
}

.dropdown:hover .dropdown-content {
  display: block;
}
```

Listing 5: Dropdown Menu CSS Implementation

4.4 Responsive Behavior

The navigation bar adapts to different screen sizes through media queries:

- On small screens, the TUM text logo becomes visually hidden.
- The dropdown positioning adjusts for mobile devices reach.
- Touch targets are sized appropriately for mobile interaction.

```
@media screen and (max-width: 767px) {
  .navbar {
    margin-top: auto;
  }

  .nav-links {
    position: absolute;
    right: 1rem;
  }

  .page__header-logo {
    left: 0;
  }

  .page__header-text {
    color: transparent;
  }
}
```

Listing 6: Navigation Responsive CSS

5 Hero Section (Ahmed)

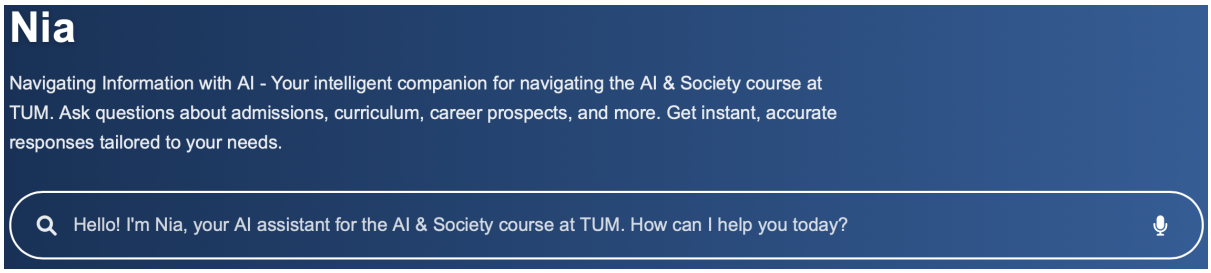

Figure 4: Hero Section

5.1 Hero Section Design

The hero section serves as the primary entry point for user interaction, featuring:

- A heading introducing "Nia," the AI assistant.
- Descriptive text explaining the purpose of the chatbot.
- A search input field for chat interactions.
- Voice input capability.

The visual design uses a gradient background in TUM blues to mimic the main website:

```
.hero {
  background: linear-gradient(90deg, #072140 0%, #3070B3 100%);
  padding: 3rem 0;
  min-height: fit-content;
  padding-top: calc(60px + 3rem);
  display: flex;
  align-items: flex-start;
  justify-content: center;
  transition: min-height var(--transition-normal);
}

.hero.expanded {
  min-height: fit-content;
  height: auto;
}
```

Listing 7: Hero Section Styling

5.2 Search Input Implementation

The search input component combines multiple functionalities:

- Text input field for typing questions.
- Voice input button for speech-to-text functionality.
- Visual feedback during interaction.

```
<div class="search-container">
    <div class="search-wrapper">
        <i class="fas fa-search search-icon" aria-hidden="true"></i>
        <label for="userInput" class="visually-hidden">Ask a question</label>
        <input type="text" id="userInput" class="search-input"
               placeholder="Hello! I'm Nia, your AI assistant for the AI & Society course at TUM. How can I help you today?"
               aria-label="Ask a question about AI & Society course">
        <button class="voice-input" aria-label="Use voice input">
            <i class="fas fa-microphone" aria-hidden="true"></i>
        </button>
    </div>
</div>
```

Listing 8: Search Input HTML

5.3 Chat Window Implementation

Figure 5: Chat Window Interface

The chat window is the core interactive component, featuring:

- Message history display with visual distinction between user and bot messages.
- Typing indicators that provide feedback during response generation.
- Chat management buttons for clearing or downloading the conversation.
- Smooth scrolling behavior to follow new messages.

The chat window which is initially hidden and appears only after user interacts with it

```
function showChatWindow() {
    heroSection.classList.add('expanded');
    chatWindow.classList.add('active');

    // Ensure messages fit
    const chatHeader = chatWindow.querySelector('.chat-header');
```

```
    const headerHeight = chatHeader.offsetHeight;
    chatMessages.style.maxHeight = `calc(100% - ${headerHeight}px)`;

    if (!chatWindow.classList.contains('active')) {
        // Expand hero section first
        heroSection.classList.add('expanded');

        // Show chat window with animation
        setTimeout(() => {
            chatWindow.classList.add('active');
        }, 300);

        // Smooth scroll to chat window on mobile
        if (window.innerWidth <= 768) {
            setTimeout(() => {
                const chatSectionTop = document.getElementById('chat-section').offsetTop;
                window.scrollTo({ top: chatSectionTop, behavior: 'smooth' });
            }, 100);
        }
    }
}
```

Listing 9: Chat Window Toggle Implementation

5.4 Message Handling

Messages in the chat interface are dynamically created and styled:

```
function addMessage(message, isUser = false) {
    const messageDiv = document.createElement('div');
    messageDiv.classList.add('message');
    messageDiv.classList.add(isUser ? 'user-message' : 'bot-message');

    const textSpan = document.createElement('span');
    // Use textContent for security, but handle line breaks
    const formattedMessage = message.replace(/\n/g, '<br>');
    textSpan.innerHTML = formattedMessage;
    messageDiv.appendChild(textSpan);
```

```
11
12      // Add role attributes for accessibility
13      messageDiv.setAttribute('role', 'listitem');
14      messageDiv.setAttribute('aria-label', `${isUser ? 'You' : '
            Assistant'}: ${message}`);
15
16      chatMessages.appendChild(messageDiv);
17
18      // Ensure proper scroll after adding message
19      requestAnimationFrame(() => {
20          chatMessages.scrollTo({
21              top: chatMessages.scrollHeight,
22              behavior: 'smooth'
23          });
24      });
25  }
```

Listing 10: Message Creation Implementation

5.5 Voice Input Integration

The voice input functionality uses the Web Speech API for speech recognition:

```
1   if ('SpeechRecognition' in window || 'webkitSpeechRecognition' in
        window) {
2       const SpeechRecognition = window.SpeechRecognition || window.
            webkitSpeechRecognition;
3       const recognition = new SpeechRecognition();
4
5       recognition.continuous = false;
6       recognition.interimResults = false;
7       recognition.lang = 'en-US';
8
9       let isListening = false;
10
11      voiceInput.addEventListener('click', () => {
12          if (!isListening) {
13              recognition.start();
14              voiceInput.querySelector('i').classList.remove('fa-
                    microphone');
15              voiceInput.querySelector('i').classList.add('fa-
                    microphone-slash');
```

```
            voiceInput.setAttribute('aria-label', 'Listening... Click to stop');

            // Add visual feedback
            searchInput.placeholder = "Listening...";
            voiceInput.classList.add('listening');
        } else {
            recognition.stop();
            voiceInput.querySelector('i').classList.remove('fa-microphone-slash');
            voiceInput.querySelector('i').classList.add('fa-microphone');
            voiceInput.setAttribute('aria-label', 'Use voice input');

            // Reset feedback
            searchInput.placeholder = "Hello! I'm Nia, your AI assistant for the AI & Society course at TUM. How can I help you today?";
            voiceInput.classList.remove('listening');
        }
        isListening = !isListening;
    });
}
```

Listing 11: Voice Input Implementation

6 FAQ Section (Hangze)

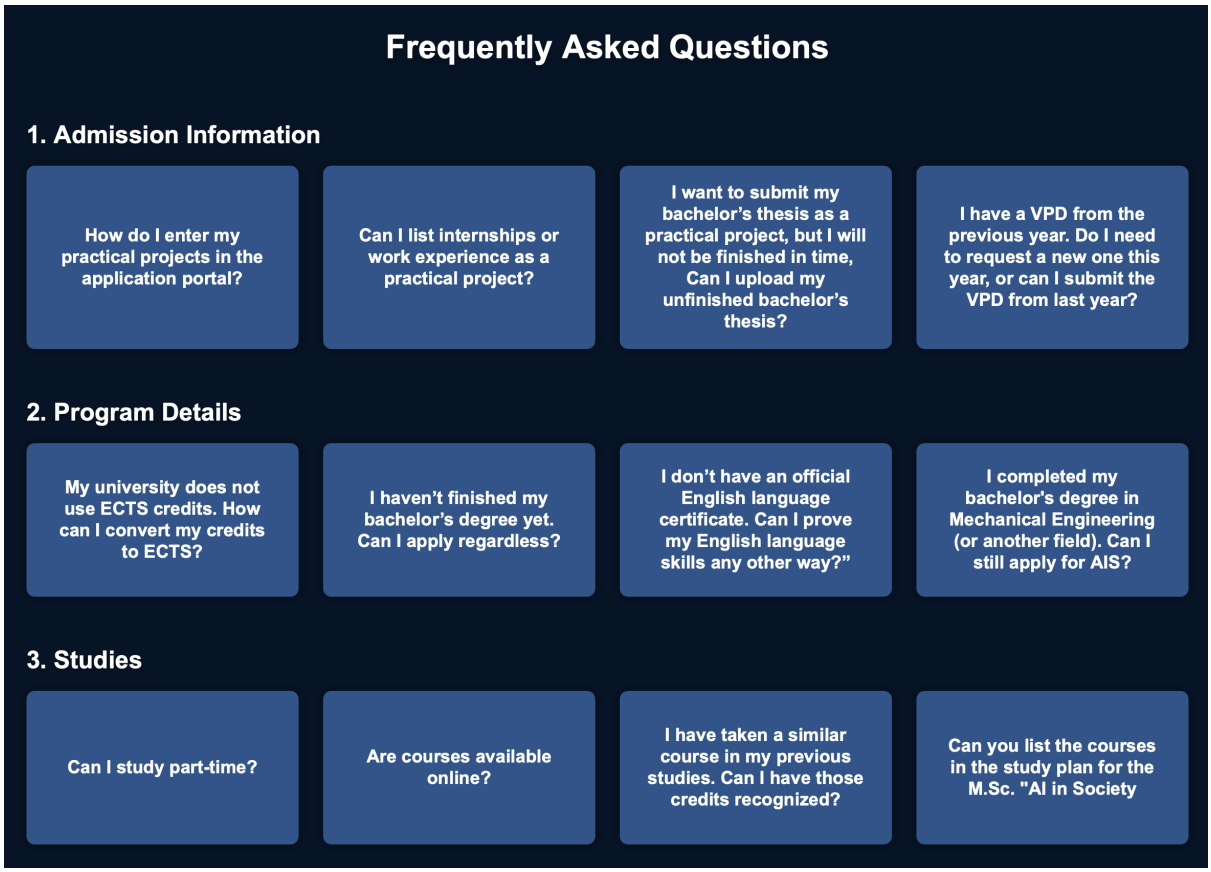

Figure 6: FAQ section

6.1 Design and Structure

The FAQ section is designed to provide quick access to common questions, organized in a grid layout:

```
<section class="faq-section">
    <div class="section-container">
        <h2>Frequently Asked Questions</h2>

        <!-- Section 1: Admission Information -->
        <h3 class="faq-subtitle">1. Admission Information</h3>
        <div class="faq-grid">
            <div class="faq-item" tabindex="0" role="button" aria
               -label="Click to ask about required documents for
               admission">
                <h3>How do I enter my practical projects in the
                   application portal?</h3>
```

```
            </div>
        </div>
    </div>
</section>
```

Listing 12: FAQ Section HTML

Key structural elements include:

- Section heading identifying the purpose
- Category subheadings for organization
- Interactive question items with proper accessibility attributes
- Grid layout for visual organization

6.2 Styling Implementation

The FAQ section uses a distinctive styling approach to create visual interest:

```
.faq-section {
  height: auto;
  padding: 4rem 0;
  background-color: var(--tum-blue);
}

.faq-grid {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 3rem 1.5rem;
  margin-bottom: 3rem;
  min-height: 150px;
}

.faq-item {
  background: var(--tum-light-blue);
  padding: 1rem 1.5rem;
  border-radius: 0.5rem;
  cursor: pointer;
  box-shadow: 0 0 7px 3px rgba(0, 0, 0, 0.2);
  transition: transform var(--transition-normal), box-shadow var
     (--transition-normal);
  text-align: center;
```

```
  display: flex;
  align-items: center;
  justify-content: center;
  width: 100%;
  height: 100%;
  position: relative;
  overflow: hidden;
}

.faq-item:hover {
  transform: translateY(-5px);
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.25);
}
```

Listing 13: FAQ Section Styling

Notable styling features include:

- Grid-based layout for visual organization.
- Shadow effects to create depth.
- Hover interactions with transform effects.
- Responsive adjustments for different screen sizes.

6.3 Interactive Functionality

The FAQ items allow users to trigger the chatbot with pre-defined questions:

```
faqItems.forEach(item => {
    item.addEventListener('click', () => {
        const question = item.querySelector('h3').textContent;
        handleInput(question);
    });

    // Add keyboard support for accessibility
    item.addEventListener('keydown', (e) => {
        if (e.key === 'Enter' || e.key === ' ') {
            e.preventDefault();
            const question = item.querySelector('h3').textContent
                ;
            handleInput(question);
        }
```

```
    });
});
```

Listing 14: FAQ Interaction JavaScript

Key functionality includes:

- Click handling to extract and submit the question text
- Integration with the main chat input handling system
- Visual feedback through CSS transitions

6.4 Responsive Adaptation

The FAQ grid adapts to different screen sizes through media queries:

```
@media (max-width: 768px) {
  .faq-section h2 {
    font-size: 1.8rem;
  }

  .faq-grid {
    grid-template-columns: repeat(2, 1fr);
    gap: 2rem 1rem;
  }
}

@media (max-width: 480px) {
  .faq-section h2 {
    font-size: 1.5rem;
  }

  .faq-grid {
    grid-template-columns: repeat(2, 1fr);
    gap: 1.5rem 0.5rem;
  }

  .faq-item h3 {
    font-size: 1rem;
  }
}
```

Listing 15: FAQ Responsive CSS

7 Footer Section (Jiayi)

Figure 7: Footer section

7.1 Design and Structure

The footer section provides comprehensive institutional information and important links.

```
<footer class="page__footer">
    <div class="page__footer-primary">
        <div class="footer_top">
            <div class="flex">
                <!-- TUM Logo and contact information -->
                <div class="flex__md-4">...</div>

                <!-- Support section -->
                <div class="flex__md-4">...</div>

                <!-- Study coordinator information -->
                <div class="Footer_text flex__md-4">...</div>

                <!-- Schools and quicklinks -->
                <div class="footer_links">...</div>
            </div>
        </div>
    </div>
    <div class="page__footer-tertiary">
        <div class="footer_bottum">
            <!-- Secondary links and social media -->
```

```
            <div class="quick_social flex">...</div>

            <!-- Partners section -->
            <div class="Footer_text partners-container">...</div>
        </div>
    </div>
</footer>
```

Listing 16: Footer HTML Structure (Simplified)

Key structural elements include:

- Flexible grid system for content organization.
- Appropriate link grouping for related content.

7.2 Content Organization

The footer content is organized into logical sections:

- **Institutional Information**: TUM logo, address, and contact details.
- **Support Section**: Information on supporting the university.
- **Study Coordinator**: Contact information for program-specific inquiries.
- **Academic Schools**: Links to other schools within TUM.
- **Quick Links**: Takes to university resources.
- **Secondary Links**: Policy, accessibility, and emergency information.
- **Social Media**: Linking to TUM's social platforms.
- **Partners**: Corporate partners.

7.3 Styling Approach

The footer styling implements a two-tone approach to divide two seperate sections:

```
.page__footer-primary {
  background-color: var(--tum-blue-secondary);
}

.page__footer-tertiary {
  width: 100%;
  background-color: var(--tum-blueish);
```

```
}

.footer_top {
  display: flex;
  justify-content: space-between;
  padding: 2rem;
  max-width: 1200px;
  background-color: var(--tum-blue-secondary);
  margin: 0 auto;
}

.footer_bottum {
  max-width: 1200px;
  padding: 2rem;
  font-size: 0.7rem;
  background-color: var(--tum-blueish);
  margin: 0 auto;
}
```

Listing 17: Footer Styling CSS

7.4 Responsive Implementation

The footer adapts to different screen sizes through flexible layouts and media:

```
@media (max-width: 768px) {
  .flex__md-4 {
    flex: 1 1 100%;
  }

  .page__footer-quicklinks ul {
    gap: 5px;
  }

  .page__footer-social-media ul {
    justify-content: center;
  }

  .footer-links {
    justify-content: center;
    text-align: center;
  }
```

```
}
```

Listing 18: Footer Responsive CSS

8 Pop-Ups

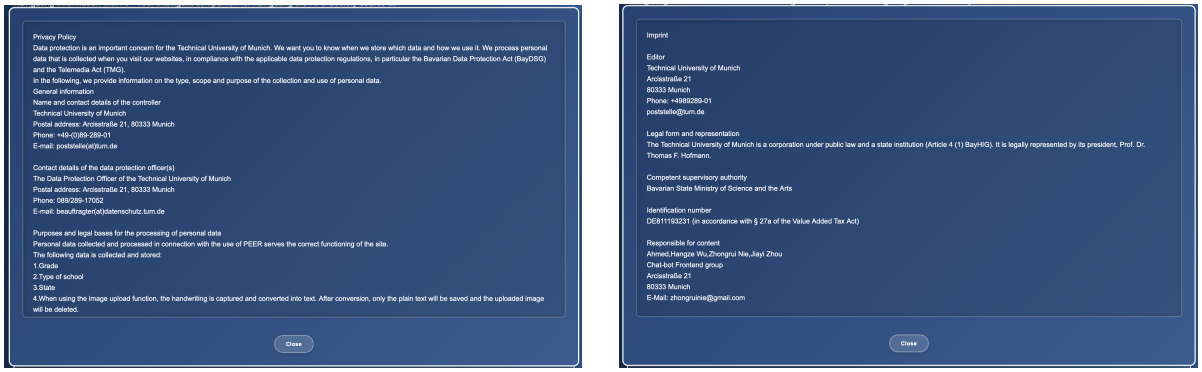

Figure 8: Pop-Ups

- **Privacy Policy**: Displays comprehensive privacy information
- **Contact Information**: Provides contact details and institutional information

These Pop-Upa use a combination of HTML, CSS, and JavaScript:

```
window.togglePrivacyPolicy = function() {
    const policyBox = document.getElementById("privacyContainer");
    if (policyBox.style.display === "none" || policyBox.style.display === "") {
        policyBox.style.display = "block";
    } else {
        policyBox.style.display = "none";
    }
};

window.closePrivacyPolicy = function() {
    document.getElementById("privacyContainer").style.display = "none";
};
```

Listing 19: Modal Dialog JavaScript Implementation

9 Cookie Consent (Ahmed)

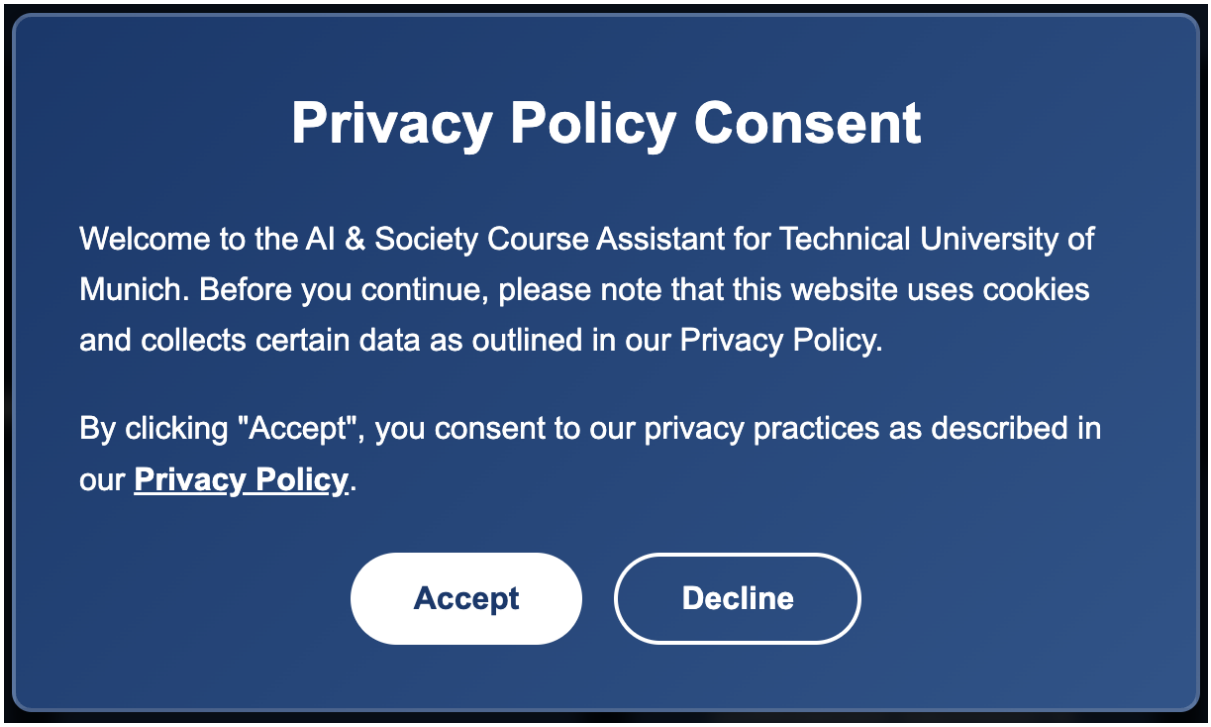

Figure 9: Cookie Pop-Up

9.1 Regulatory Compliance

In accordance with the General Data Protection Regulation (GDPR) and the ePrivacy Directive, the website implements a cookie consent mechanism to inform users about data collection practices and obtain explicit consent before storing non-essential cookies.

Our implantation does not collect any user data at all. Although the implementation felt necessary in case of storing users chats is to be implemented.

- The consent mechanism is presented before any interactions.
- Detailed information about cookie usage is available in the privacy policy with one click.
- It encompasses a responsive implementation making sure that the cookie consent mechanism remains usable across devices.
- The cookie consent popup is designed to be clearly visible to users when they first open the page:

10 Conclusion and Future Work

10.1 Future Enhancements

Several potential enhancements could further improve the frontend:

- **Multi-language Support**: Adding language switching capabilities for international students and german speakers.
- **Analytics**: Implementing usage tracking to identify common questions.
- **User Feedback System**: Adding mechanisms for the users to provide feedback.
- **Responsiveness Improvement**: The project was made as part of a course, but many features can be simply improved providing an expert touch.

10.2 Conclusion

The AI in Society Course Assistant frontend team was able to represents a successful implementation of all the tasks required to integrate the chat-bob, By focusing on a solid foundation for student interaction with the AI chatbot.

As the AI in Society program evolves, this frontend architecture provides a flexible framework that can be changed according to requirements.