Toothbook.ca System Design Document

Authors:

* Arifali Baghirli - 210201848
* Zoher Matar - 210201919
* Yahye Said Abdi - 210201922
* Bashir Tayani - 220201812
* Abdulbari Abdulghani Ahmed Leftesi - 220201877
* Ramiz Shafiyev - 220201988
* Bashar Ali Mohamed Osman - 220201863
* Ahmed Mohamed Ahmed – 200201874

Table of Contents:

**1. Introduction**

* 1.1 Purpose of the Document
* 1.2 Scope of the System
* 1.3 Overview

**2. System Architecture**

* 2.1 High-Level Architecture Diagram
* 2.2 Technology Stack
* 2.3 Module Breakdown
  + Appointment Module
  + Clinic Management Module
  + Review Module
  + User Module
  + Admin Panel
  + Notification System

**3. Component Design**

* 3.1 Data Flow Diagrams (DFD)
* 3.2 UML Diagrams
  + Use Case Diagram
  + Sequence Diagram
  + Class Diagram
* 3.3 Database Design (ER Diagram + table definitions)

**4. Interface Design**

* Screens or mockups
* API endpoint structure (sample request/response)

**5. Security Design**

* Authentication & Authorization (JWT, role-based access)
* Data protection (HTTPS, encryption)

**1.1 Purpose of the Document**

This document defines the system architecture, module design, data models, and technology stack for the development of the Toothbook Online Clinic Appointment System. It is intended for developers, testers, and stakeholders to understand the internal design and how different components interact.

**1.2 Scope of the System**

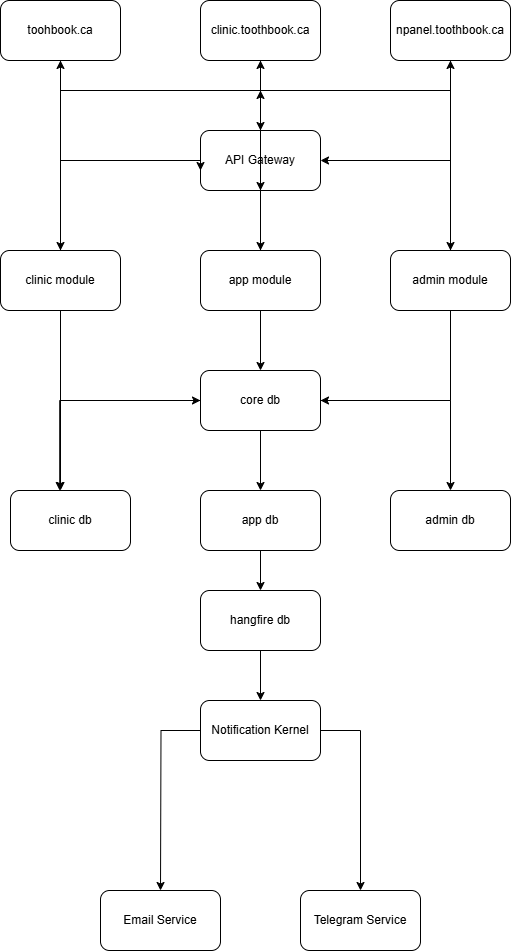
The system will allow patients to search for clinics, book appointments, manage their visits, and leave reviews. Clinics will manage their profiles, schedules, and appointments. Admins will oversee the entire platform. The system will be delivered as a responsive web application.

**1.3 Overview**

The document includes high-level architecture, component/module breakdown, data design, UI wireframes, and system interfaces. It aims to guide the full development life cycle.

**2. System Architecture**

**2.1 High-Level Architecture Diagram**



**2.2 Technology Stack**

| **Component** | **Technology** |
| --- | --- |
| Frontend | React.js + Typescript + Tailwind CSS |
| Backend | ASP.NET Core 8 Web API |
| Database | Microsoft SQL Server |
| Hosting | Windows Shared Hosting,Windows VPS,IIS Server |
| Authentication | ASP .NET Core Identity, JWT, OAuth2 |
| CI/CD | GitHub |
| Dev Tools | Visual Studio, VS Code, Postman, Figma, Adobe XD, Canva |
| Diagrams | draw.io, Lucidchart, Apex Charts |

**3. Module Breakdown**

|  |  |
| --- | --- |
| **Module** | **Description** |
| **App Module** | Handles patient requests, login, register, book, send message and etc. |
| **Clinic Module** | Handles clinic users’ requests, edit appointments, contact administration, add/remove credit cards, not to pay requests |
| **Admin Module** | Manage users, clinics, charge clinics, website contents |
| **Notification Module** | Sends automatic, manual reminders to app users, clinic users, admin users, advertisement emails, reminders about booking statuses, helps tracking booking statuses |

3. Component Design

**3.1 Data Flow Diagram (DFD) – Level 1**

[Patient]

|→ (Search Clinics) → [Clinic Module]

|→ (Book Appointment) → [App Module]

|→ (Write Review) → [App Module]

← (Get Notification) ← [Notification Kernel]

[Clinic]

|→ (Login) → [Clinic Module]

|→ (Update Schedule) → [Clinic Module]

|→ (View Appointments) → [App Module]

[Admin]

|→ (Manage Clinics/Users) → [Admin Module]

[Modules]

→ (Data) → [API Gateway]

→ (Routing) → [Relevant DBs: clinic db, app db, core db, etc.]

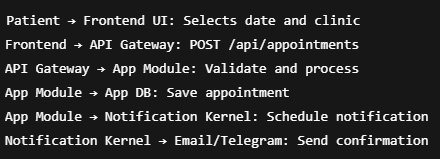
**3.2 UML Diagrams**

**a. Use Case Diagram**

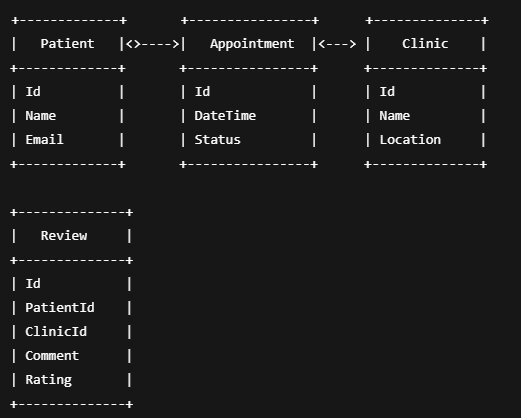
Actors: Patient, Clinic, Admin  
System: Toothbook Online Clinic Appointment System



**b. Sequence Diagram – Book Appointment**



**c. Class Diagram**



**3.3 Database Design (ER + Table Definitions)**

**a. ER Diagram Overview**

Patients --< Appointments >-- Clinics

Clinics --< Reviews >-- Patients

VisitRequests – NotificationQueue

**b. Sample Table Definitions**

**Table: Patients**

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Description** |
| Id | UUID | Primary key |
| Name | Text | Full name |
| Email | Text | Unique login |
| Password | Text | Hashed password |

**Table: Appointments**

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Description** |
| Id | UUID | PK |
| PatientId | UUID | FK to Patients |
| ClinicId | UUID | FK to Clinics |
| DateTime | Timestamp | Date & time |
| Status | Varchar | Scheduled/Done/Cancelled |

**4. Interface Design**

**4.1 Screens or Mockups (Text Description)**

**toothbook.ca (Patient)**

* Home (search clinics by location/service)
* Clinic Profile Page (info + schedule + reviews)
* Appointment Booking
* Review Form
* Login/Register

**clinic.toothbook.ca (Clinic)**

* Login/Register
* Dashboard (upcoming appointments)
* Schedule Manager
* Review Viewer

**npanel.toothbook.ca (Admin)**

* Login
* Manage Clinics
* User Monitoring
* System Logs

**4.2 API Endpoint Structure**

**Sample Request**

POST /api/Interactive/Booking/createbooking

Authorization: Bearer <token>

Content-Type: application/json

{

"clinicId": 0,

"doctorId": 0,

"year": 0,

"month": 0,

"day": 0,

"hour": 0,

"minute": 0,

"note": "string",

"iHaveBeenInClinic": true,

"logId": 0,

"entryPoint": "string"

}

**Sample Response**

{

"success": true,

"message": "Appointment booked successfully",

"appointmentId": "0"

}

**5. Security Design**

**5.1 Authentication & Authorization**

* JWT tokens used across all modules.
* Token issued after login.
* Token contains user\_id, role, and expiry.

**Roles:**

* Patient
* Clinic User
* Doctor
* Admin
* CEO
* CTO
* Sales Manager
* Marketing Manager
* Branding Manager
* Developer

Each endpoint is protected based on role and policies e.g.:

* [Authorize(Policy = "IsClinicUser")]
* [Authorize(Policy = "IsEmployee")]
* [AllowAnonymous]

**5.2 Data Protection**

* All traffic encrypted via **HTTPS,SSL/TLS,HSTS**
* Passwords and sensitive data hashed using B**crypt, SHA256 or RSA256**
* Sensitive fields (email, mobile, stripe customer credentials, credit cards, social security numbers, patient insurance numbers, clinic credit card numbers and etc.) are encrypted in DB
* Rate-limiting on API Gateway to prevent brute-force
* SQL Injection prevention via ORM (Entity Framework Core 8, Dapper)
* Cross-site scripting (XSS) protection via frontend encoding