**PrepForge**

**Vision:**

Ace your next tech interview with a personalized, AI-powered preparation journey, curated questions, real-time progress tracking, and structured study paths, all in one seamless portal.

**Why is it different from other portals or apps out there?**There are portals that *partially* overlap what you’re thinking, but none that match exactly what this portal provides, especially not with the "AI + personal journey + progress tracking" combo.

**So, not even GPT?**

**No,** GPT is a great tool but it wont provide a dashboard, or checklist, or even download the progress report of a particular user. The portal uses GPT (AI), to get the required questions, based on the specific role asked by the user, but rest of the features are provided by the portal, which helps job seekers to prep in a good way.

**Tech Stack:**

Django + Django Framework

React.js

PostgreSQL

AWS

**Link:**  
<https://chatgpt.com/share/680f1567-3c3c-8010-b43d-9fb3a76ef29a>

**Models.py:**  
This is used to create tables in the database.

By applying commands:

* python manage.py makemigrations
* python manage.py migrate

The first command tells Django, that these are the tables in the models.py, which need to be created in the database (blueprint).

The second command actually applies these tables to the database (actual development).

**Then we add a file serializers.py in interview folder:**

This file, actually creates the data in .json format, this helps the frontend to access the data when called from backend.  
You can think of serializers as:

**Translator between Django models and HTTP API data.**

**Views.py:**

This creates different instances such as role list, question list, options to create, update, delete the data.

|  |  |
| --- | --- |
| View | Purpose |
| RoleListView | List all roles (Frontend Developer, SDE-1, etc.) |
| QuestionListView | List all questions related to a specific role |
| UserProgressListCreateView | See / Create user's progress (bookmarked, completed, etc.) |
| UserProgressDetailView | Update / Delete user's progress |
| UserNoteListCreateView | See / Create user's notes for questions |
| UserNoteDetailView | Update / Delete user's notes |

**urls.py:**

It **maps URLs to views** like this:

|  |
| --- |

|  |  |  |
| --- | --- | --- |
| URL | View | Purpose |
| /roles/ | RoleListView | Fetch list of roles |
| /questions/<role\_id>/ | QuestionListView | Fetch questions by role |
| /progress/ | UserProgressListCreateView | Create/View progress records |
| /progress/<pk>/ | UserProgressDetailView | Edit/Delete progress |
| /notes/ | UserNoteListCreateView | Create/View notes |
| /notes/<pk>/ | UserNoteDetailView | Edit/Delete notes |

**Creating Superuser for backend:**

|  |
| --- |
| **Login to Admin Panel** |

|  |
| --- |
| <http://127.0.0.1:8000/admin/> |

**Username: Vedant**

**Email: k18ved@gmail.com**

**Pass: prepforge1806**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Quick Recap: What I've Built So Far:**

|  |
| --- |

|  |  |
| --- | --- |
| Part | Status |
| Django Backend Setup | ✅ |
| Models (Role, Question, Progress, Notes) | ✅ |
| Admin Panel | ✅ |
| APIs to fetch Roles/Questions | ✅ |
| Server Running Locally | ✅ |

**For Frontend:**

Had option of selecting Vite or React, and after researching I selected Vite, as it is the latest technology, faster server speed than normal React, and would also help by getting highlighted.

|  |  |
| --- | --- |
| Part | Status |
| Backend Django Server | Running on port 8000 |
| Frontend React (Vite) Server | Running on port 5173 |
| GitHub Repo | Clean — backend and frontend separated |
| Localhost Frontend | ✅ Live and ready |
| Localhost Backend API | ✅ Live and returning real data |

When getting the following error:

* **cd frontend**
* cd: no such file or directory: frontend

This most of the times, simply means you are in the same directory and don’t need to cd.

**Solve CORS (Cross Origin Request) Issue in Django (Backend):**

This issue arises, because backend is live on one host: 8000

And frontend is live on another host: 5173

So, the browser blocks them, so using **django-cors-headers** helps resolve this issue.

A screenshot of a computer

AI-generated content may be incorrect.

**"Back — let's continue from Questions Page!"**

**Quick Git commands to add, commit and push:**

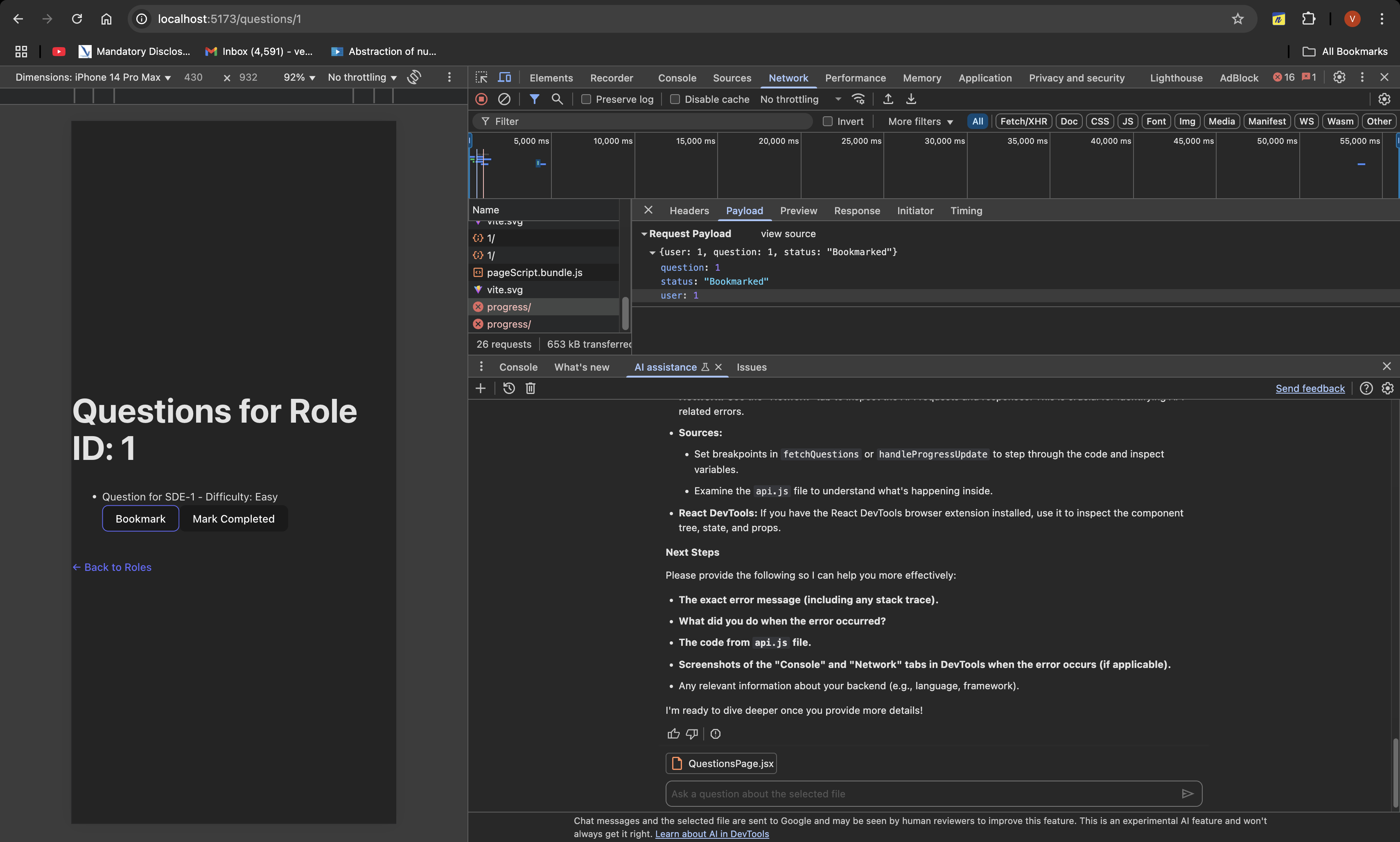
git add .

git commit -m "Meaningful message like: Setup Django models and admin panel"

git push

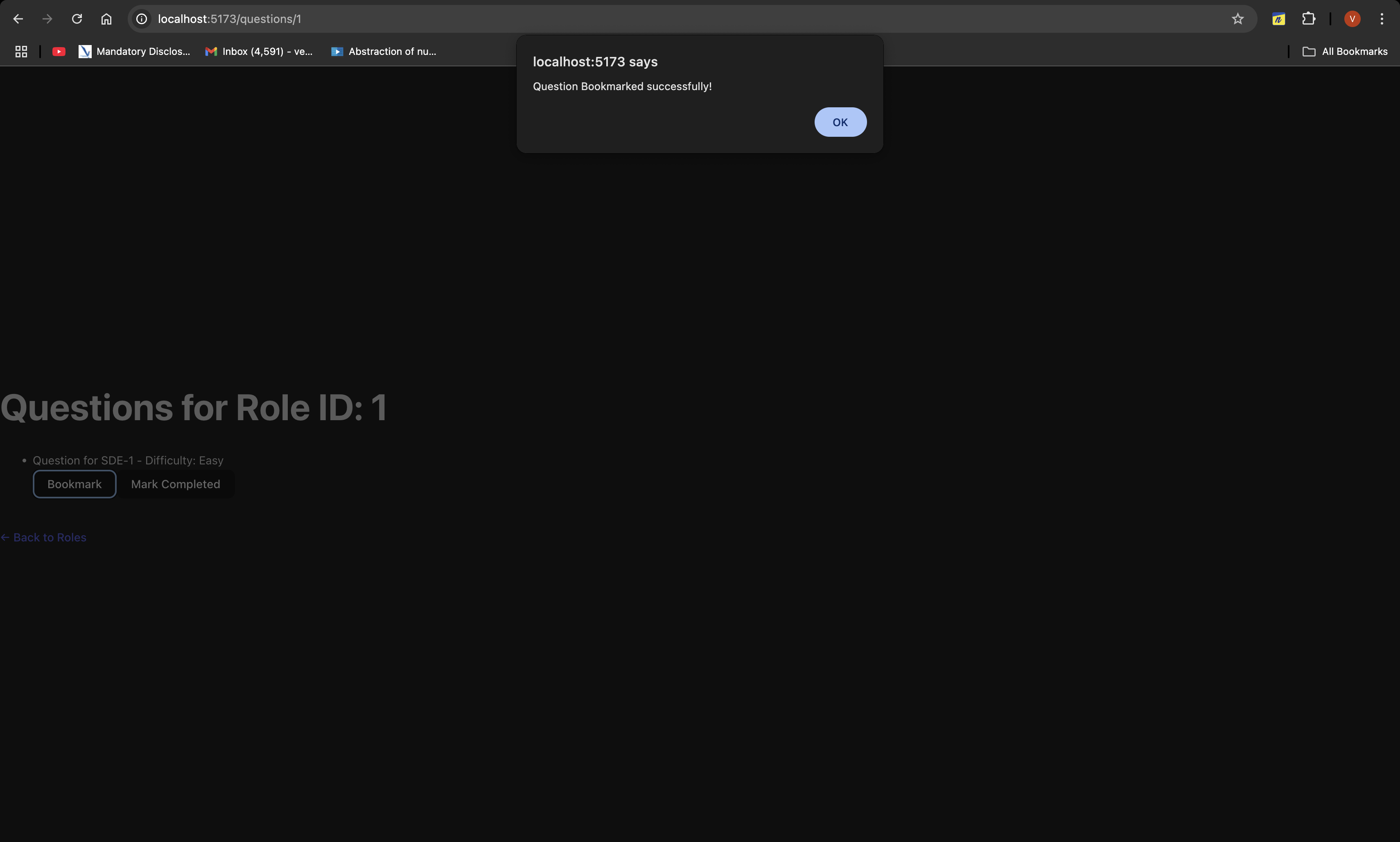
Started with the adding new feature:  
Add Book mark and Mark as Completed features, but encountered errors.

The following is the one of the way to check if frontend is working properly:  
Under Inspect => Network => Under the ‘Name’ tab, you can see the error, after clicking it, you get the option which says ‘payload’, after clicking it, you get the output, that the fronend is sending. This helps in debugging, if this is correct, there is no issue with the frontend, changes needs to be done in backend.

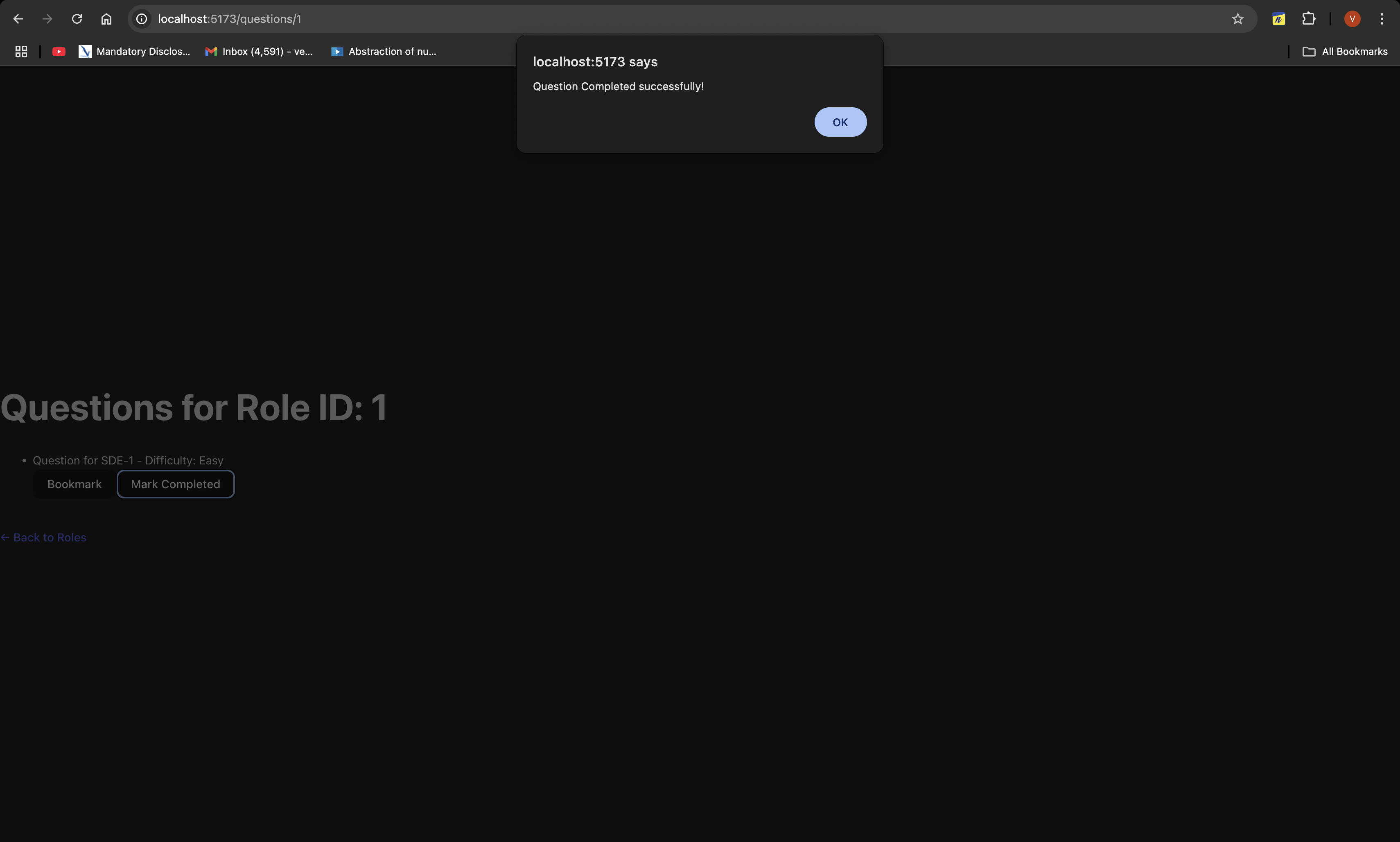


**Use of Bookmark vs Mark Completed:**

|  |  |
| --- | --- |
| Status | Real-World Meaning |
| **Bookmarked** | "I want to practice this question later" or "I want to revisit it" |
| **Completed** | "I finished solving it" or "I understand it fully" |

**Working Bookmark:  
**

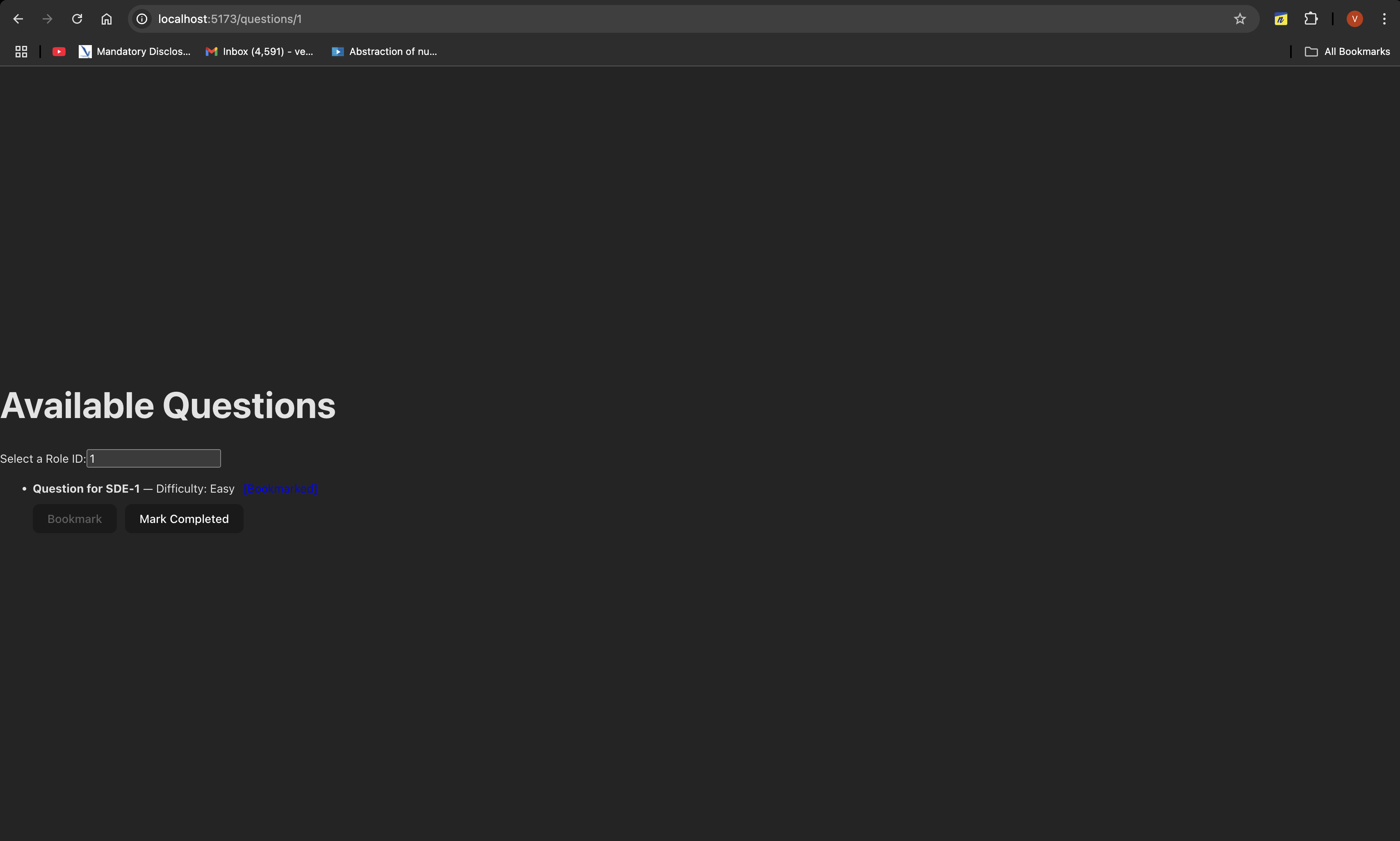
**Working Mark Completed:**

****

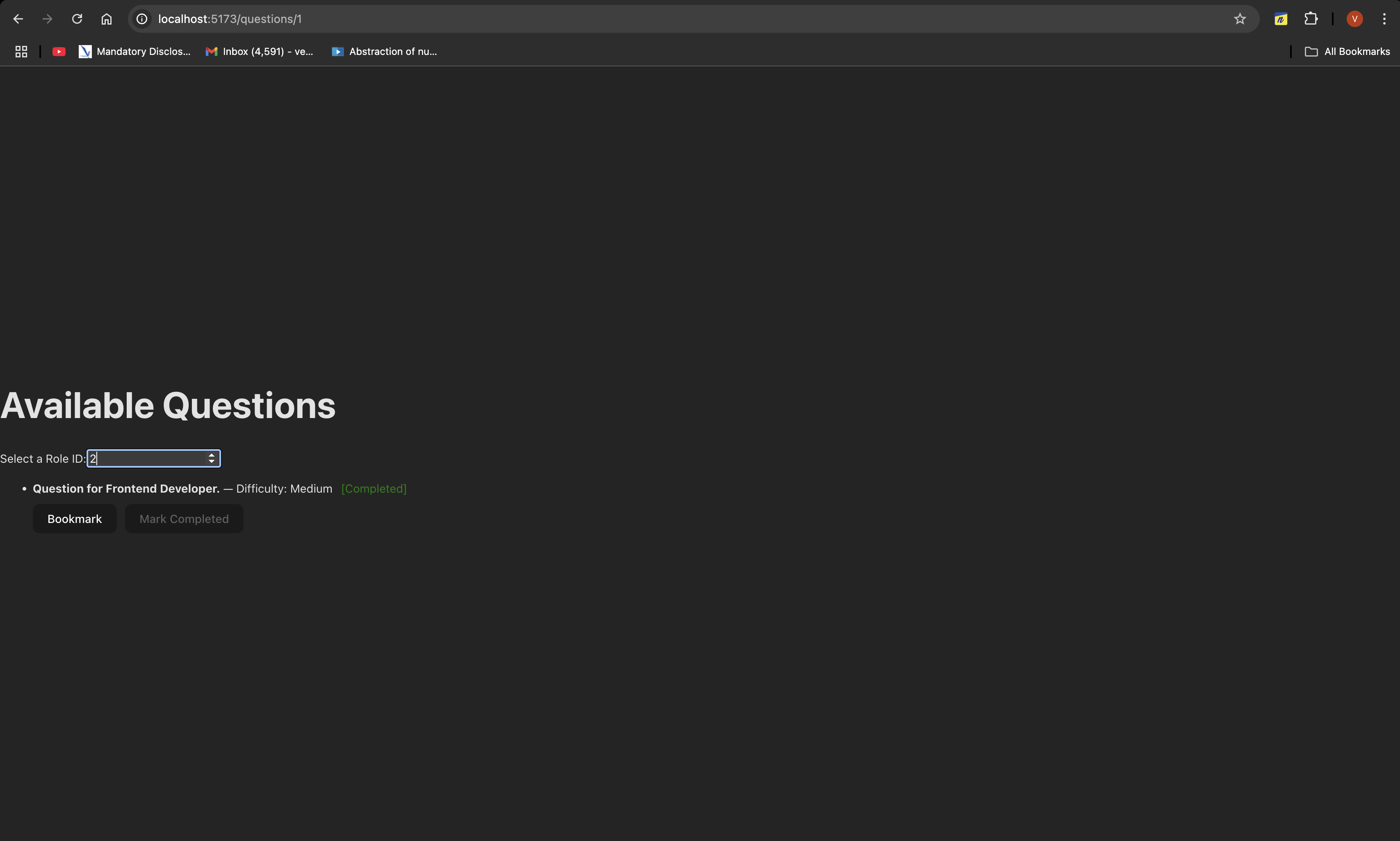
**Added the feature:**

This feature highlights what the user has selected from the two, and fades the button which is selected and highlights the button which is open to select.

In the image below, user selects Bookmark, hence it is faded and Mark Completed button is highlighted. Also, there is an option to select the role id.



In the image below, the user has selected Mark Completed, so Bookmark button is faded and Mark Completed button is highlighted, along with the, user selected input.



"**Ready to polish UI ✅ Let’s go! 🚀**"

**Adding the User Notes Feature:**

**This allows the user to add their notes for each question, like ‘need to revise this’, ‘imp question’, or write their own understanding out of it, which would truly act as a prep platform.**

**Authentication Phase:**

**1. What exactly is JWT?**

**JWT** = **JSON Web Token**

It's simply **a special string** (kind of like a digital ticket 🎫)  
✅ It proves "**Hey, I am Vedant — here's my signed proof!**"

Example JWT Token (don't worry about format):

**eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MX0.LX57NmseukPi4JS...**

**Frontend Structure:**

| **Step** | **Action** |
| --- | --- |
| 1 | Create a simple LoginPage.jsx |
| 2 | Create a form (Username + Password) |
| 3 | When user clicks Login ➔ Axios POST /api/token/ |
| 4 | Save received JWT token in localStorage |
| 5 | Redirect to Questions page after login ✅ |

**One of the few milestone achieved in building this:**

Was getting the error, which were:

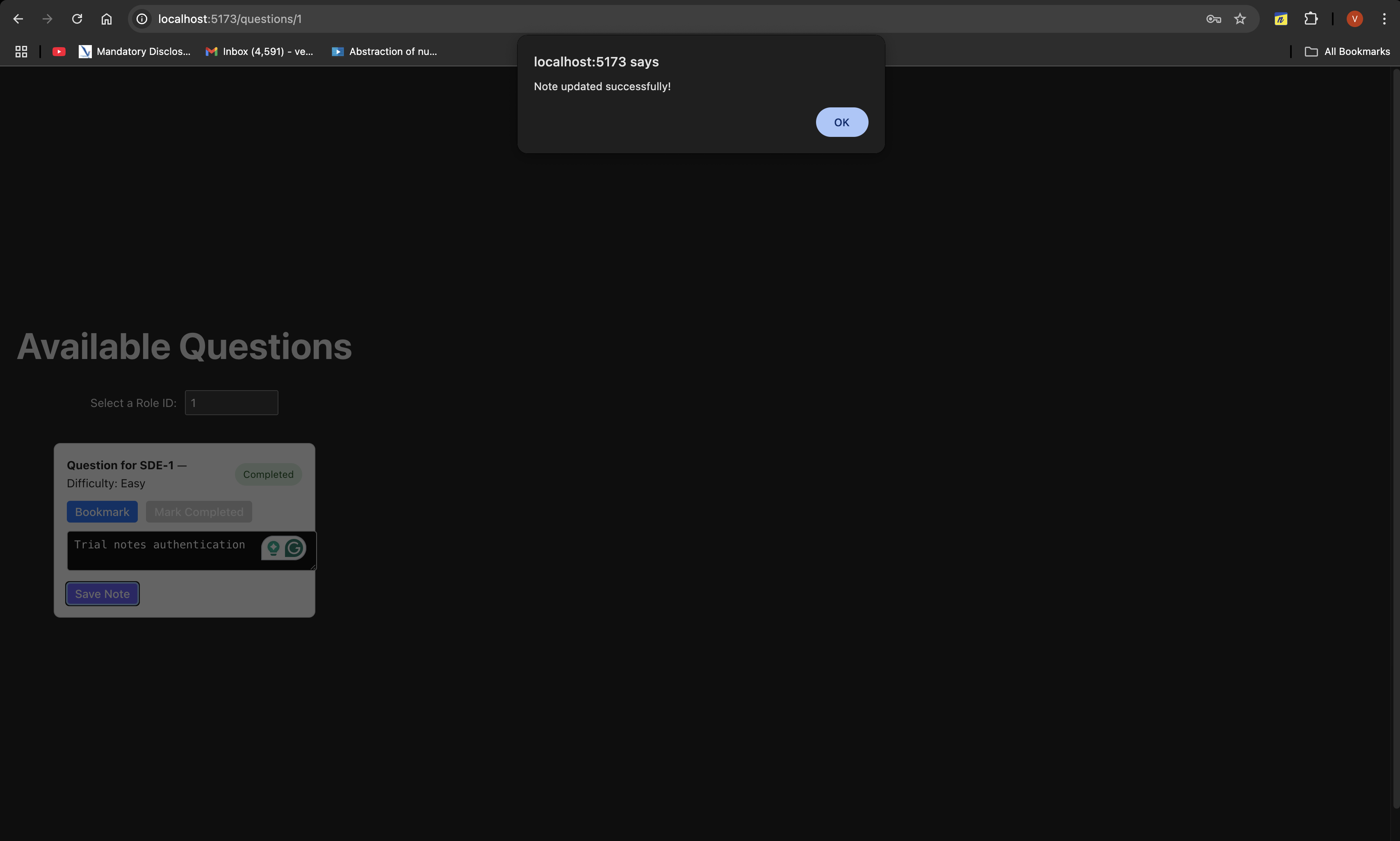
1: User Id was required before, but after JWT authentication, it wasn’t needed, so was giving errors, when user id wasn’t provided from the frontend.

Solution: Removed the user id requirement from the frontend, question.jsx file.

2: Whenever I was trying to change something in notes and save it, it used to throw error, which says, failed to save notes.

Solution: Made changes to the frontend, such that, whenever there are any changes to the notes, and the user clicks save, the alert says, Notes updated successfully.

These were the major challenges faced.



**The following is the victory map of the project till this point:**

**[Frontend: React + Vite]**

**|**

**├── Home Page (Choose Role) ➔ /questions/:roleId**

**├── Login Page (JWT Login) ➔ /login**

**└── Questions Page (Per Role)**

**├── Load Questions based on Role**

**├── Mark Questions as Bookmark/Complete**

**├── Write & Save Notes (Create or Update smartly!)**

**|**

**▼**

**[API: Axios]**

**|**

**├── GET /api/roles/**

**├── GET /api/questions/:role\_id/**

**├── POST+PATCH /api/progress/ (Bookmark/Completed status)**

**├── POST+PATCH /api/notes/ (User Notes)**

**├── JWT Authorization Header**

**|**

**▼**

**[Backend: Django REST Framework (DRF)]**

**|**

**├── Models**

**│ ├── Role**

**│ ├── Question**

**│ ├── UserProgress**

**│ └── UserNote**

**├── Serializers**

**│ ├── RoleSerializer**

**│ ├── QuestionSerializer**

**│ ├── UserProgressSerializer (user auto-attached)**

**│ └── UserNoteSerializer (user auto-attached)**

**├── Views**

**│ ├── ListAPIView (Role, Question)**

**│ ├── ListCreateAPIView (Progress, Note)**

**│ └── JWT Protected APIs**

**├── JWT Authentication System**

**|**

**▼**

**[Database: SQLite]**

**└── Stores Users, Roles, Questions, Progress, Notes**

**Before vs After — Saving Notes:**

|  |
| --- |

|  |  |  |
| --- | --- | --- |
| Action | Before Fix | After Fix |
| Saving new note | **✅ Works (POST)** | **✅ Works (POST)** |
| Saving again for same question | **❌ "unique set" error** | **✅ Smart PATCH update** |
| User experience | **❌ Feels broken** | **✅ Smooth and production-grade** |
| Backend behavior | **❌ Always POST, caused duplication** | **✅ POST if new, PATCH if exists** |
| Database integrity | **❌ Risky** | **✅ Super clean** |
|  |  |  |

**What We Can Build Next:**

* **Signup System (Register New Users)**
* **Toast Notifications for Save Success/Fail**
* **Profile Page: View My Bookmarks, Completed**
* **Role Dropdown UX Improvement**
* **AWS Deployment (putting whole app live!)**
* **Invite-Only Beta Mode (make it feel premium)**

**Built signup page:**

**Contains:**

**Username, Email, Password, Confirm Password**

After clicking submit button it redirects to the login page. Here user can login using recently signed up account.