New project confirmed.. building detailed architecture! 💻💡

### \*\*Website Architecture and Pseudocode Plan\*\* for Local 701:

---

### \*\*1. User Authentication (Login System)\*\*

\*\*Objective\*\*: Implement a secure login system with tracking for user activity.

#### \*\*Backend Flow:\*\*

- \*\*Database\*\*: Store user credentials (hashed passwords), user roles (admin/member), and activity logs.

- \*\*APIs\*\*:

- `POST /api/auth/login`: Validate credentials and create a session token.

- `POST /api/auth/register`: Register a new member.

- `POST /api/auth/logout`: Invalidate session token.

- `GET /api/auth/session`: Check active session.

#### \*\*Pseudocode:\*\*

```python

# Pseudocode for Login system

def login(username, password):

# Check if username exists

user = find\_user\_by\_username(username)

if user and verify\_password(user.password, password):

# Log successful login and return session token

token = generate\_session\_token(user)

log\_user\_activity(user, "Login")

return token

else:

raise InvalidCredentialsError

def logout(user):

# Invalidate session token

invalidate\_session\_token(user.token)

log\_user\_activity(user, "Logout")

# Function to track user activity

def log\_user\_activity(user, action):

current\_time = get\_current\_time()

ip\_address = get\_user\_ip\_address()

save\_to\_activity\_log(user.id, action, current\_time, ip\_address)

```

#### \*\*Frontend\*\*:

- \*\*Login Page\*\* with fields for username and password.

- \*\*Member Dashboard\*\* after login (redirect).

---

### \*\*2. User Activity Tracking\*\*

\*\*Objective\*\*: Track the pages users visit and actions they take (e.g., downloads, posts).

#### \*\*Backend Flow\*\*:

- \*\*Database\*\*: Store logs with timestamp, user ID, action (e.g., “downloaded document”, “posted message”), and IP.

- \*\*APIs\*\*:

- `POST /api/activity/log`: Log user activity whenever they visit a page or take specific actions.

#### \*\*Pseudocode:\*\*

```python

# Pseudocode for activity tracking

def track\_page\_visit(user, page):

current\_time = get\_current\_time()

log\_user\_activity(user, f"Visited {page}")

def track\_action(user, action):

current\_time = get\_current\_time()

log\_user\_activity(user, action)

# Example: When user downloads a document

def download\_document(user, document\_id):

document = get\_document\_by\_id(document\_id)

log\_user\_activity(user, f"Downloaded document {document.name}")

return document

```

#### \*\*Frontend\*\*:

- \*\*Tracking events\*\* like button clicks (e.g., downloading a document) or navigation actions (e.g., visiting the forum).

---

### \*\*3. Discussion Board\*\*

\*\*Objective\*\*: Provide a forum for members to engage in discussions under various categories.

#### \*\*Backend Flow\*\*:

- \*\*Database\*\*: Tables for threads, posts, categories, and replies.

- \*\*APIs\*\*:

- `GET /api/forum/categories`: Get list of categories.

- `POST /api/forum/create-thread`: Create a new thread.

- `POST /api/forum/post-reply`: Post a reply to a thread.

- `GET /api/forum/thread/{id}`: Get the details of a specific thread.

#### \*\*Pseudocode:\*\*

```python

# Pseudocode for creating a forum thread

def create\_thread(user, category\_id, title, content):

thread = Thread(

user\_id=user.id,

category\_id=category\_id,

title=title,

content=content,

created\_at=get\_current\_time()

)

save\_to\_database(thread)

log\_user\_activity(user, f"Created thread {title}")

# Pseudocode for posting a reply

def post\_reply(user, thread\_id, content):

reply = Reply(

user\_id=user.id,

thread\_id=thread\_id,

content=content,

created\_at=get\_current\_time()

)

save\_to\_database(reply)

log\_user\_activity(user, f"Posted reply in thread {thread\_id}")

```

#### \*\*Frontend\*\*:

- \*\*Forum Interface\*\*: Threads organized by categories.

- \*\*Post/reply functionality\*\*.

---

### \*\*4. Document Management\*\*

\*\*Objective\*\*: Allow members to access, download, and possibly sign/view documents.

#### \*\*Backend Flow\*\*:

- \*\*Database\*\*: Store documents with metadata (e.g., file type, category, date uploaded).

- \*\*APIs\*\*:

- `GET /api/documents`: Retrieve list of documents (filtered by category).

- `POST /api/documents/upload`: Allow admins to upload new documents.

- `GET /api/documents/{id}/download`: Download a specific document.

#### \*\*Pseudocode:\*\*

```python

# Pseudocode for document management

def list\_documents(user):

# Filter by category if needed

documents = get\_all\_documents()

log\_user\_activity(user, "Viewed document list")

return documents

def download\_document(user, document\_id):

document = get\_document\_by\_id(document\_id)

log\_user\_activity(user, f"Downloaded document {document.name}")

return download(document.file\_path)

```

#### \*\*Frontend\*\*:

- \*\*Document List Page\*\*: Organized by categories.

- \*\*Download functionality\*\*.

---

### \*\*5. News and Event Updates\*\*

\*\*Objective\*\*: Display latest news and upcoming events on the homepage.

#### \*\*Backend Flow\*\*:

- \*\*Database\*\*: Store news articles, event details, and registration links.

- \*\*APIs\*\*:

- `GET /api/news`: Fetch news articles.

- `GET /api/events`: Fetch event details.

#### \*\*Pseudocode:\*\*

```python

# Pseudocode for fetching news and events

def get\_latest\_news():

news\_list = get\_news\_from\_database()

return news\_list

def get\_upcoming\_events():

events\_list = get\_events\_from\_database()

return events\_list

```

#### \*\*Frontend\*\*:

- \*\*News Section on Homepage\*\*: Preview of latest news with links to full articles.

- \*\*Event Section\*\*: Details of upcoming events with registration links.

---

### \*\*6. Security and Responsive Design\*\*

\*\*Objective\*\*: Secure the site with HTTPS and ensure it looks great on both desktop and mobile devices.

#### \*\*Security\*\*:

- \*\*Enable HTTPS\*\* using SSL certificates.

- \*\*Encrypt sensitive data\*\* (e.g., passwords).

#### \*\*Responsive Design\*\*:

- Use \*\*media queries\*\* in CSS to ensure the website adapts to different screen sizes.

---

### \*\*Directory Structure\*\*:

```plaintext

/ (root)

│

├── /src

│ ├── /auth # Authentication services

│ ├── /forum # Discussion board

│ ├── /documents # Document management

│ ├── /news # News & events

│ ├── /tracking # User activity tracking

│ └── /common # Common utilities

│

├── /public

│ ├── /assets # Images, stylesheets, fonts

│ ├── index.html # Homepage

│ ├── about.html # About Us page

│ ├── dashboard.html # Member Dashboard

│ └── forum.html # Forum page

│

└── /styles

├── main.css # Main styles

└── responsive.css # Responsive styles for mobile

```

---

Let me know if you'd like to proceed with implementation, or if any adjustments are needed! 👾