Taf'Yaa's Design document

Layer	Tool/Stack	Purpose
Frontend	React.js/CSS	UI and interaction
Tree Display	D3.js	Tree graph rendering
Backend	Firebase(for DB, Auth and Role Acceess control), Netlify(for cloud functions at the MVP level)	Manage backend logic
Database	Firestore (NoSQL)	Family + person + relations
File Storage	Cloudinary	Audio, photos
Hosting	to be decided but the present options are Vercel or Firebase + Netifl	App hosting
Export	jsPDF / html2canvas	PDF/image tree snapshot export

Database Schema Design (Collection Overview)

Collection Name	Purpose / Description
`Users`	Stores user account data including email, profile, language preferences, etc.
`Members`	Represents individual family members in the tree with personal and relational data.
`Stories`	Contains audio or text stories associated with members or branches of the tree.
`Media`	Stores uploaded audio files, images, and videos, usually referenced by stories or members.
`Trees`	Contains metadata and configuration for a specific family tree.
`Invitations`	Handles user invites via email or QR code for collaborative tree editing.
`JoinRequests`	When a user wants to join a tree, this collection logs their request and status.
`Notifications`	Stores in-app notifications and alerts for users.
`Suggestions`	Stores AI-generated relationship or correction suggestions for members.
`publicMathPool`	Stores AI-generated relationship or correction suggestions for members.

1. `people` Collection


```
{
  "id": "p001",
  "treeId": "tree001",
  "name": "Chief Tanda",
  "gender": "male",
  "dob": "1935-06-20",
  "dod": null,
  "photoUrl": "https://cloudinary.com/chief_tanda.jpg",
  "bio": "A respected elder from Bafang...",
  "language": "bafang",
  "fatherId": null,
  "motherId": null,
  "spouseIds": ["p010", "p011"],
  "childrenIds": ["p100", "p101"],
  "linkedUserId": null,
  "isDeceased": true,
  "publicConsent": true,
  "updatedAt": "2025-07-09T12:00:00Z"
}
```



```
{
  "id": "p001",
  "treeId": "tree001",
  "name": "Chief Tanda",
  "gender": "male",
  "dob": "1935-06-20",
  "dod": null,
  "photoUrl": "https://cloudinary.com/chief_tanda.jpg",
  "bio": "A respected elder from Bafang...",
  "language": "bafang",
  "fatherId": null,
  "motherId": null,
  "spouseIds": ["p010", "p011"],
  "childrenIds": ["p100", "p101"],
  "linkedUserId": null,
  "isDeceased": true,
  "publicConsent": true,
  "createdAt": "2022-07-09T12:00:00Z"
  "updatedAt": "2025-07-09T12:00:00Z"
}
```


2. `trees` Collection

```
{
  "id": "tree001",
  "familyName": "Tanda Dynasty",
  "createdBy": "user_musa_001",
  "createdAt": "2024-06-01T10:15:00Z",
  "language": "bafang",
  "isPublic": false,
  "memberCount": 36,
  "currentRootId": "p001",
  "invitesEnabled": true,
  "mergeOptIn": true
}
```

3. `users` Collection

```
{
  "uid": "user_musa_001",
  "email": "musa@gmail.com",
  "displayName": "Musa Tanda",
  "profilePhoto": "https://cloudinary.com/musa.jpg",
  "linkedPersonId": "p100",

  "joinedTrees": ["tree001", "tree002"],

  "roles": {
    "tree001": "admin",
    "tree002": "viewer"
  },

  "preferences": {
    "language": "bafang",
    "darkMode": true,
    "treeDefaultView": "radial"
  },

  "invitedBy": "user_sister_01",
  "lastLogin": "2025-06-30T08:20:00Z"
}
```

4. `stories` Collection

```
{
  "storyId": "story001",
  "treeId": "tree001",
```

```
  "personId": "p001",
  "title": "The Palm War",
  "type": "audio", // or "text"
  "language": "bafang",
  "text": null,
  "audioUrl": "https://cloudinary.com/story001.mp3",
  "addedBy": "user_musa_001",
  "timestamp": "2024-06-15T13:20:00Z",
  "tags": ["clan conflict", "oral", "village"]
}
```

5. `events` Collection

```
{
  "eventId": "evt001",
  "treeId": "tree001",
  "personId": "p100",
  "title": "Marriage to Amina",
  "type": "marriage",
  "date": "1962-04-12",
  "description": "A joyful union in Bamenda"
}
```

6. `invites` Collection

```
{
  "code": "XVBDRAfou024",
  "treeId": "tree001",
  "createdBy": "user_musa_001",
  "uses": 10,
  "role": "member",
  "expiresAt": "2025-01-01T00:00:00Z",
  "status": "active", // Or expired
 }
```

7. `media` Collection

```
{
  "mediaId": "media01",
  "treeId": "tree001",
  "ownerType": "person", // person or story
  "ownerId": "p001",
  "fileType": "audio",
  "url": "https://cloudinary.com/story001.mp3",
```

```
  "uploadedBy": "user_musa_001",
  "timestamp": "2024-06-01T12:00:00Z",
  "tag": "story", // or "profile"
  "language": "bafang"
}
```

---

8. `publicMatchPool`

```
{
  "publicMatchId": "match001",
  "personId": "p001",
  "treeId": "tree001",
  "name": "Chief Tanda",
  "dob": "1935-06-20",
  "fatherName": "Kouma",
  "motherName": "Ndeya",
  "village": "Bafang",
  "isLinkedToRoot": true,
  "optedInBy": "user_musa_001",
  "language": "bafang"
}
```

9. `notifications` Collection

```
{
  "id": "notif_001",
  "userId": "user_musa_001",
  "type": "invite",              // "invite" | "joinRequest" | "system" | "media" | "comment"
  "title": "You were invited to join Tanda Dynasty",
  "message": "Click to accept the invitation from your cousin.",
  "treeId": "tree001",
  "relatedId": "XVBDRAfou024",   // Could be inviteId, joinRequestId, etc.
  "isRead": false,
  "createdAt": "2025-08-04T10:23:00Z"
}
```

10. `joinRequests` Collection

```
{
  "requestId": "jr001",
  "treeId": "tree001",
  "requestedBy": "user_jane_007",
```

```
  "message": "Hi! I believe this is my great-grandfather.",
  "status": "pending",         // "pending" | "accepted" | "rejected"
  "reviewedBy": null,          // filled when status changes
  "reviewedAt": null,
  "createdAt": "2025-08-04T10:30:00Z"
}
```

API Specification

Authentication & Users

Method	Path	Description	Auth Required	Request Body / Params	Response
GET	`/me`	Get current user profile	Yes	Token (header)	User JSON
POST	`/link-user`	Link user account to a personId	Yes	`{ personId }`	Success JSON

Trees

Method	Path	Description	Auth Required	Request Body / Params	Response
GET	`/trees/:id`	Get tree metadata	depends on visibility	`id` (URL param)	Tree JSON
POST	`/trees`	Create a new tree	Yes	Tree payload	Tree JSON
POST	`/trees/:id/merge`	Request merge with another tree	Yes (admin)	`{ targetTreeId }`	Merge status
GET	`/trees/:id/members`	Get members in tree	Yes	Tree ID	Array of people
GET	`/trees/:id/stories`	Get stories for tree	Yes	Tree ID	Array of stories

People / Members

Method	Path	Description	Auth Required	Request Body / Params	Response
GET	`/members/:id`	Get person info	Yes	`id` (URL param)	Person JSON

POST	`/members`	Create new person	Yes	Person payload	Person JSON
PATCH	`/members/:id`	Update person	Yes	Partial update payload	Person JSON
DELETE	`/members/:id`	Delete person	Yes (admin)		Success

Stories

Method	Path	Description	Auth Required	Request Body / Params	Response
GET	`/stories/:id`	Get story detail	Yes	`id` (URL param)	Story JSON
POST	`/stories`	Add new story	Yes	Story payload	Story JSON
DELETE	`/stories/:id`	Delete story	Yes (owner/admin)		Success

Media

Method	Path	Description	Auth Required	Request Body / Params	Response
POST	`/media/upload`	Upload file (audio/image)	Yes	Multipart file + metadata	Media URL JSON
GET	`/media/:id`	Stream/download media	Yes	mediaId param	Media Blob

Notifications

Method	Path	Description	Auth Required	Request Body / Params	Response
GET	`/notifications`	Get user notifications	Yes		Array of notifications
PATCH	`/notifications/:id`	Mark as read	Yes		Updated notification

Join Requests

Method	Path	Description	Auth Required	Request Body / Params	Response
POST	`/join-requests`	Request to join tree	Yes	`{ treeId, message }`	Join request JSON
PATCH	`/join-requests/:id`	Accept/reject request	Yes (admin)	`{ status }`	Updated request

Since we are using Firebase as both the backend and database, we have intentionally omitted several API endpoints, especially those related to user authentication and management.

These functionalities are securely handled client-side using the Firebase SDK, which provides methods for user registration, login, profile updates, management.

Functionality	Firebase Feature Used	Reason for Excluding Custom API
User Registration	`createUserWithEmailAndPassword()`	Handled by Firebase Authentication SDK client-side; no backend needed.
User Login	`signInWithEmailAndPassword()`	Secure login handled by Firebase Auth.
User Logout	`signOut()`	Directly handled via SDK on client.
Update User Profile	`updateProfile()`	Firebase Auth provides this natively.
Delete User Account	`auth.currentUser.delete()`	No need for custom API endpoint.
Get Current User Info	`auth.currentUser` or `onAuthStateChanged()`	Real-time auth state listener from Firebase SDK.

Detailed UI/UX wireframes

Authentication & Onboarding

No	UI Name	Purpose
1	Welcome / Landing Page	App introduction and entry point
2	Onboarding Tutorial	Step-by-step initial user guidance
3	Login	Firebase authentication
4	Signup	New user registration
5	Create Family Tree	Initialize new family tree structure
6	Join Family Tree (QR/Code)	Access via shared invite code or QR
7	Account Settings	Manage profile, language, and logout

Tree Exploration & Visualization

No	UI Name	Purpose
8	Navigation Bar	Global access to core sections
9	Tree View (Canvas)	Interactive family tree visualization
10	Switch Root	Change perspective ancestor
11	Tree Export	Download tree as PDF/image
12	Export Success Modal	Confirmation after successful export
13	Tree Not Found	Error handling for invalid tree access
14	Public Search	Discover people from public trees
15	Tree Member Profile View	Full tree member profile with relationships

Profiles & Content Management

No	UI Name	Purpose
16	Add/Edit Member	Create/modify person profiles
17	Upload Oral Story	Record/attach audio narratives
18	Add Life Event	Log milestones (birth, marriage, death)
19	Media Preview Modal	Preview images/audio before publishing
20	User Profile Page	Account details and preferences

AI & Relationships

No	UI Name	Purpose
21	Suggestion List	AI-generated relationship matches
22	Suggestion Detail	Evidence/reasoning for matches
23	Propose Merge	Initiate tree merge request
24	Merge Request Inbox	Manage incoming merge proposals
25	Merge History	Track accepted/rejected merges

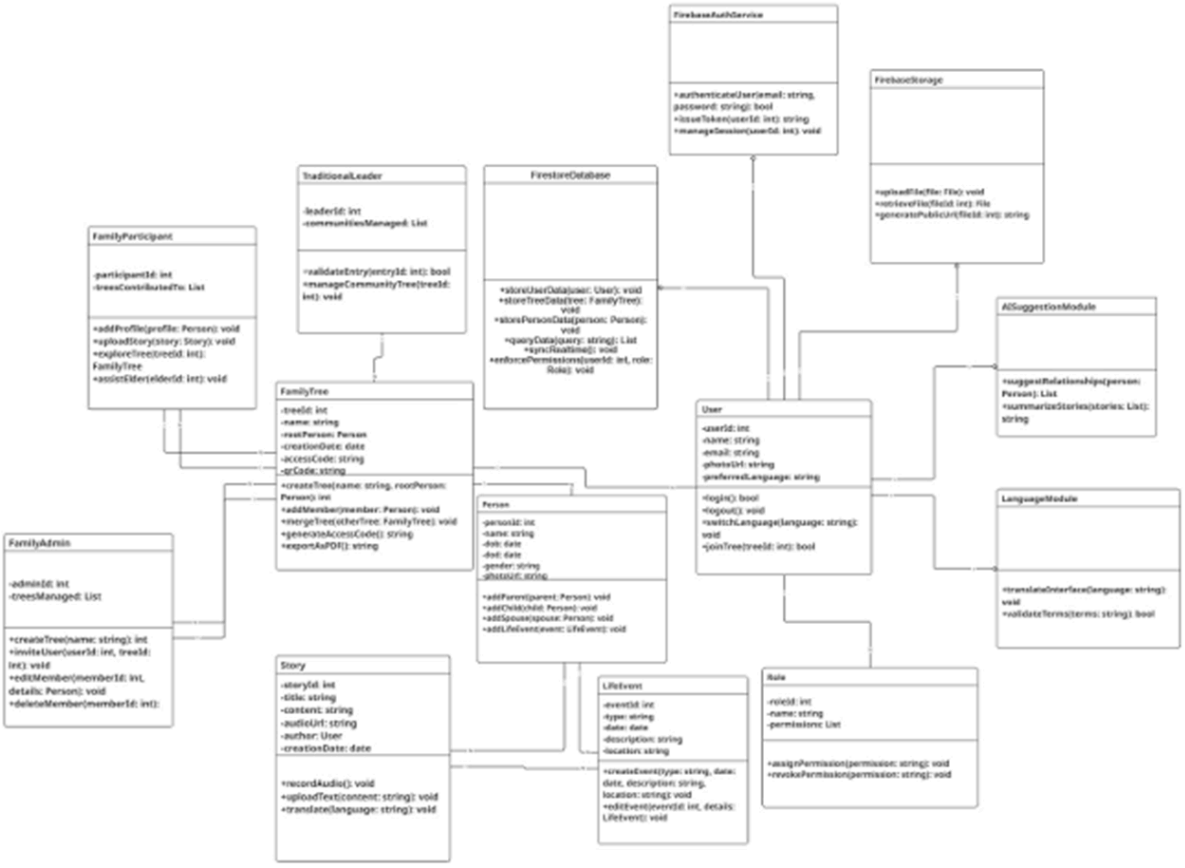
Access Control & Administration

No	UI Name	Purpose
26	Invite Members	Generate/share tree access codes
27	Invite Code Modal	Quick role assignment during onboarding
28	Role Management	Assign/edit user permissions
29	Merge Confirmation	Approve/reject merge requests (modal)
30	Tree Settings	Configure privacy and export defaults

Support & Feedback

No	UI Name	Purpose
31	Help & Glossary	Terminology and feature explanations
32	Notification Hub	Centralized alerts for merges/invites/actions
33	Feedback Popup	Report issues/suggest improvements (modal)

Class Diagram


FirebaseAuthService
+authenticate(user/email: string, password: string): bool
+issueToken(userId: int): string
+manageSession(userId: int): void
FirebaseStorage
+uploadFile(file: File): void
+retrieveFile(fileId int): file
+generatePublicUrl(fileId: int): string
TraditionalLeader
-leaderId: int
-communitiesManaged: List
+validateEntry(entryId: int): bool
+manageCommunityTree(treeId: int): void
FirestoreDatabase
+storeUserData(user: User): void
+storeTreeData(tree: FamilyTree): void
+storePersonData(person: Person): void
+queryData(query: string): List
+sync(Realtime(): void
+enforcePermissions(userId: int, role: Role): void
FamilyParticipant
-participantId: int
-treesContributedTo: List
+addProfile(profile: Person): void
+uploadStory(story: Story): void
+exploreTree(treeId: int): FamilyTree
+assistElder(elderId: int): void
AISuggestionModule
+suggestRelationships(person: Person): List
+summarizeStories(stories: Lis(): string
FamilyTree
-treeId: int
-name: string
-rootPerson: Person
-creationDate: date
-accessCode: string
-qrCode: string
+createTree(name: string, rootPerson: Person): int
+addMember(member: Person): void
+mergeTree(otherTree: FamilyTree): void
+generateAccessCode(): string
+exportAsPDF(): string
User
-userId: int
-name: string
-email: string
-photoUrl: string
-preferredLanguage: string
+login(): bool
+logout(): void
+switchLanguage(language: string): void
+joinTree(treeId: int): bool
LanguageModule
+translateInterface(language: string): void
+validateTerms(terms: string): bool
FamilyAdmin
-adminId: int
-treesManaged: List
+createTree(name: string): int
+inviteUser(userId: int, treeId: int): void
+editMember(memberId: int, details: Person): void
+deleteMember(memberId: int):
Person
-personId: int
-name: string
-dob: date
-dod: date
-gender: string
-photoUrl: string
+addParent(parent: Person): void
+addChild(child: Person): void
+addSpouse(spouse: Person): void
+addLifeEvent(event: LifeEvent): void
Story
-storyId: int
-title: string
-content: string
-audioUrl: string
-author: User
-creationDate: date
+recordAudio(): void
+uploadText(content: string): void
+translate(language: string): void
LifeEvent
-eventId: int
-type: string
-date: date
-description: string
-location: string
+createEvent(type: string, date: date, description: string, location: string): void
+editEvent(eventId: int, details: LifeEvent): void
Role
-roleId: int
-name: string
-permissions: List
+assignPermission(permission: string): void
+revokePermission(permission: string): void