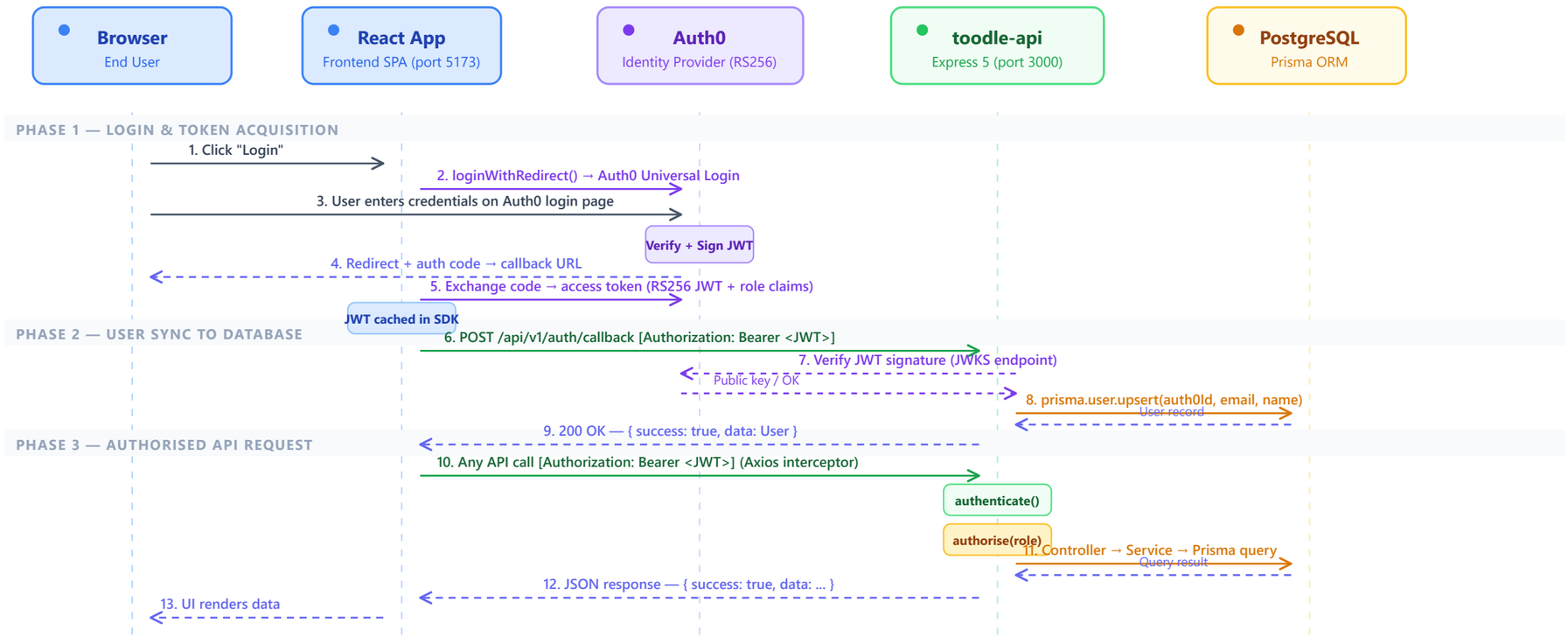
Toodle — Authentication & Authorisation Flow

Auth0 Universal Login · JWT RS256 · RBAC middleware · User sync · Dev bypass



Phase 1 — Login & Token Acquisition

#	ACTOR	ACTION	DETAIL
1	User → Browser	Click Login button	`LandingPage` / `LoginButton` calls `useAuth().login()`
2	React App → Auth0	`loginWithRedirect()`	Auth0 SDK redirects browser to Universal Login page with PKCE code challenge; `appState.returnTo` preserves current path
3	User → Auth0	Submit credentials	Email + password on Auth0 hosted login page
4	Auth0 → Browser	Redirect + auth code	Auth0 redirects to app callback URL; SDK automatically exchanges code for tokens
5	Auth0 → React App	Issue RS256 JWT access token	Token includes custom claim `{"{"}audience{"}"}/roles: ["ORGANISER"]` ; cached inside Auth0 SDK (memory + optional cookie)

Phase 2 — User Sync to Database

#	ACTOR	ACTION	DETAIL
6	React App → API	`POST /api/v1/auth/callback`	Axios interceptor (`setupAuthInterceptor`) automatically attaches `Authorization: Bearer <token>` ; called once after login
7	API → Auth0	JWT signature verification	`express-oauth2-jwt-bearer` fetches JWKS from `https://<domain>/.well-known/jwks.json` ; validates `iss` , `aud` , `exp` , RS256 signature

#	ACTOR	ACTION	DETAIL
8	API → PostgreSQL	`prisma.user.upsert()`	Creates user on first login; updates `email` + `name` on subsequent logins; key: `auth0Id` (Auth0 `sub` claim)
9	API → React App	Return user object	`{ success: true, data: User }` — frontend stores profile in context/state

Phase 3 — Authorised API Request

#	ACTOR	ACTION	DETAIL
10	React App → API	Any authenticated request	Axios interceptor calls `getAccessTokenSilently()` ; injects `Authorization: Bearer <JWT>` on every request; 401/403 responses logged client-side
–	API Middleware	`authenticate()`	`express-oauth2-jwt-bearer` validates JWT; populates `req.auth.payload` with decoded claims including roles
–	API Middleware	`authorise(...roles)`	Reads `req.auth.payload["{"{"}audience{"}"}"/roles"]` ; returns `403 Forbidden` if role not in allowed list
11	API	Controller → Service → Prisma	Business logic executes; Zod validators run before controllers on mutating routes
12–13	API → React App → Browser	Return JSON response	`{ success: true, data: ... }` ; React renders updated UI

RBAC Role Permissions Matrix

ROUTE	ORGANISER	TUTOR	STUDENT
`GET /auth/me`	✓	✓	✓
`POST /auth/callback`	✓	✓	✓
`GET /users`	✓	✗	✗
`GET /courses` , `GET /courses/:id`	✓	✓	✓
`POST/PATCH/DELETE /courses`	✓	✗	✗
`GET /tutors/:id`	✓	✓	✗
`GET /tutors` , `POST /tutors/:id/marks`	✓	✗	✗
`PUT /tutors/:id/availability`	✓	✓ (own)	✗
`GET/POST/PATCH/DELETE /allocations`	✓	✗	✗
`GET /allocations/validate`	✓	✗	✗

Dev Mode Bypass: The `useAuth` hook includes a `loginAsDevRole(role)` function that stores a mock user in `localStorage` (key: `toodle_dev_user`) and skips Auth0 entirely. This is for local development only — the mock user returns a static `"dev-mock-jwt-token"` instead of a real JWT, meaning backend routes will still reject requests unless the API is also running in a permissive dev mode.