- Research suggests that a modular three-tier architecture—frontend, middleware, and Microsoft 365 backend—provides the best balance of security, scalability, and maintainability for custom portals.
- Evidence leans toward using Azure AD's On-Behalf-Of (OBO) flow for middleware authentication to preserve user context while accessing Graph API, ensuring least-privilege access.
- It seems likely that integrating SharePoint for files, Teams for collaboration, and Outlook for scheduling via Graph will enable seamless internal-external access, though careful permission scoping is needed to avoid overexposure.
- The MVP should prioritize core endpoints from the provided API specification, focusing on hubs, proposals, and documents to validate the portal concept.
- While M365 Copilot and AI agents can interact via Graph, initial implementation should use delegated permissions to maintain user accountability.

Architecture Overview

The system adopts a Zero Trust model, verifying every request explicitly with least-privilege access. The frontend (given React app) handles UI and calls the middleware API. Middleware acts as a secure proxy, implementing business logic and using OBO to call Microsoft Graph. Data resides in M365 services: SharePoint/OneDrive for files, Teams for chats/meetings, Outlook for emails/calendars. This allows humans, Copilot, workflows, and agents to access the same data directly or via the portal, with the frontend easily re-skinnable via themes/CSS. learn.microsoft.com learn.microsoft.com

Key Components

- **Frontend**: React/TS app from repomix, customizable UI "skins" for different portals (e.g., proposal hub).
- **Middleware**: Azure Functions or App Service implementing REST API endpoints from API_SPECIFICATION.md, handling auth, pagination, and Graph interactions. learn.microsoft.com
- **Backend**: Microsoft Graph API for unified access to M365 data.
- **Auth**: Azure AD with separate app registrations for frontend and middleware.

Security Considerations

Implement OBO flow to avoid token relay risks, validate all tokens, handle errors like 429 throttling with retries, and apply Zero Trust pillars: explicit verification, least privilege, and assume breach. Use delegated permissions only for user-context actions. Encrypt data in transit, monitor with Defender XDR. learn.microsoft.com learn.microsoft.com

MVP Focus

Start with core features: hub creation/management, proposal uploads, document sharing. Expand to videos, messages, meetings, questionnaires. Use mock data initially for testing, then integrate Graph.

The proposed architecture for this M365-based personal SaaS product emphasizes modularity, security, and integration with existing Microsoft services to create a customizable client portal. This setup allows service companies to host portals within their M365 tenant, enabling internal teams (humans, Copilot, workflows, AI agents) to collaborate on the same underlying data as external clients. The middleware layer serves as the critical bridge, enforcing business rules and security while abstracting Graph API complexities. The first "skin" aligns with the provided repomix codebase, focusing on a proposal/project portal with features like hub overviews, document management, and client access controls. learn.microsoft.com

Detailed Architecture

The system follows a three-tier model:

1. **Presentation Tier (Frontend)**: The React/TypeScript application from the repomix (e.g., src/App.tsx, pages like HubDetail.tsx, components like ClientHubLayout.tsx). It's designed for easy re-skinning through Tailwind CSS (tailwind.config.ts) and component modularity. Features include responsive UIs for overviews, proposals, videos, documents, messages, meetings, questionnaires, and people sections. It authenticates via MSAL Browser, acquiring tokens only for the middleware scope (e.g., api://agentflow-backend/access_as_user), and never directly requests Graph scopes. ~~learn.microsoft.com~~
2. **Application Tier (Middleware)**: A serverless Azure Functions app (or Node.js/.NET App Service for stateful needs) implementing the REST API defined in docs/API_SPECIFICATION.md. It handles requests from the frontend, performs OBO token exchange to access Graph, and manages business logic like pagination, filtering, and error handling. For each "hub" (client portal), create a dedicated SharePoint site collection for files/documents/proposals, a Teams team for chats/meetings, and Outlook resources for calendars/emails. This ensures data isolation and leverages M365's built-in collaboration tools. AI agents and Copilot can interact via Graph directly, while the middleware provides a unified API for the portal. ~~learn.microsoft.com~~ ~~learn.microsoft.com~~
3. **Data Tier (M365 Backend)**: Relies on Microsoft Graph for all operations—e.g., SharePoint/OneDrive for files (proposals as PPTX in sites), Teams for messages/meetings, Outlook for emails/calendars. Data is stored in the company's M365 tenant, ensuring compliance and direct access for internal tools. ~~apriorit.com~~

Modularity is achieved by:

- Decoupling UI from logic: Frontend uses hooks (e.g., src/hooks/use-hubs.ts) to call middleware endpoints.
- Extensible API: Middleware endpoints are grouped (e.g., /hubs, /proposals) for easy addition of new features.
- Re-skinning: Theme via components.json and tailwind.config.ts; swap layouts for different portals.

Security integrates Zero Trust: learn.microsoft.com

- **Verify Explicitly**: Authenticate every API call with token validation.
- **Least Privilege**: Use delegated permissions (e.g., User.Read, Files.ReadWrite) scoped per endpoint.
- **Assume Breach**: Segment data by hub, monitor with Defender XDR, encrypt transit with TLS 1.2+.

Performance optimizations include Graph best practices: Use $select for minimal data, batch requests, delta queries for changes, and handle throttling with Retry-After. learn.microsoft.com

High-Level Product Requirements Documents (PRDs)

These PRDs outline the MVP middleware, building on the API spec. They are phased: Core (hubs, auth), Content (proposals/documents), Collaboration (messages/meetings), Analytics (activity/engagement).

1. Authentication and Authorization PRD

Objective: Secure user access with OBO flow.
Requirements:

- Register apps: Frontend (SPA), Middleware (Web API), Graph (as downstream).
- Implement OBO: Middleware validates frontend token, exchanges for Graph token using client secret/certificate.
- RBAC: Staff full access, clients read-only for their hub.
- Error Handling: 401 for invalid tokens, 403 for forbidden actions.
  Dependencies: Azure AD, MSAL.
  Metrics: 99.9% auth success rate.

2. Hub Management PRD

Objective: Create/manage client portals (hubs).
Requirements:

- Endpoints: GET/POST/PATCH /hubs as per spec.
- On create: Provision SharePoint site, Teams team, link via Graph.
- Portal Config: Toggle sections (proposal, videos, etc.), publish for clients.
- Pagination: Support page, pageSize, sort, filter.

  Dependencies: Graph APIs for sites/teams.

  Metrics: Hub creation in <10s.

3. Proposal and Document Management PRD

Objective: Handle uploads, views, engagement tracking.

Requirements:

- Upload: Multipart form to SharePoint, generate embed/download URLs.
- Engagement: Track views/time via Graph analytics.
- Versions: Maintain history in SharePoint.
- Bulk Actions: Delete/set visibility.

  Dependencies: Graph DriveItems, Analytics.

  Metrics: Upload success >99%.

4. Collaboration Features PRD

Objective: Integrate messages, meetings, questionnaires.

Requirements:

- Messages: Threaded via Teams channels.
- Meetings: Schedule via Outlook/Teams.
- Questionnaires: Link external or store in SharePoint.
- Activity Feed: Paginated events from Graph.

  Dependencies: Graph for chats/events.

  Metrics: Real-time updates via subscriptions.

5. Analytics and Monitoring PRD

Objective: Provide engagement stats and alerts.

Requirements:

- Endpoints: GET /hubs/{id}/overview, /activity.
- Track: Views, time spent, downloads via Graph insights.
- Alerts: Push via webhooks or polling.
  Dependencies: Graph Insights API.
  Metrics: Low-latency queries (<500ms).

Component	Technology	Responsibilities	Integration Points
Frontend	React/TS	UI rendering, user input, API calls	Middleware REST API
Middleware	Azure Functions (.NET/Node)	Business logic, OBO auth, Graph proxy	Azure AD, Microsoft Graph
Auth	Azure AD/MSAL	Token issuance/validation	OBO flow for Graph
Storage	SharePoint/OneDrive	Files, proposals, documents	Graph Drive API
Collaboration	Teams/Outlook	Messages, meetings, calendars	Graph Chat/Events API
Monitoring	Defender XDR	Threat detection, logs	Graph Audit Logs

This architecture ensures the portal is simple (minimal custom code), clean (Graph-centric), secure (Zero Trust), and modular (easy extensions). For implementation, start with auth setup, then core endpoints, testing against mock Graph responses.

Key Citations:

- Microsoft identity platform and OAuth2.0 On-Behalf-Of flow learn.microsoft.com
- Best practices for working with Microsoft Graph learn.microsoft.com
- Zero Trust deployment plan with Microsoft 365 learn.microsoft.com
- Best practices for working with Microsoft Graph learn.microsoft.com
- How to Secure Microsoft 365 Data with Graph API apriorit.com
- Microsoft Teams IT architecture and voice solutions posters learn.microsoft.com