Team#: 8

Team Project Title: SubClear

	First Name	Last Name	CSUSM Email	Contribution Percentage
1	Minh	Bui		
2	Bryan	Lam		
3	Kevin			
4	Braylon	S		
5				

Grading Rubrics (for instructor only):

Criteria	1. Beginning	2. Developing	3. Proficient	4. Exemplary
Use Case Diagram	0-16	16-26	27-34	35-40
	many use cases and relations are not correct	many use cases or relations are not correct	A few use cases or relations are not correct	Diagram is complete, all relations are correct
Summary level use case description table	0-5	6-9	10-14	15-20
	Missing important elements	Information provided is insufficient	Some minor issues	Information provided is sufficient and appropriate
Use case description tables for primary task use cases	0-16	16-26	27-34	35-40
	Missing important elements	Information provided is insufficient	Some minor issues	Information provided is sufficient and appropriate
Total Grade (100)				

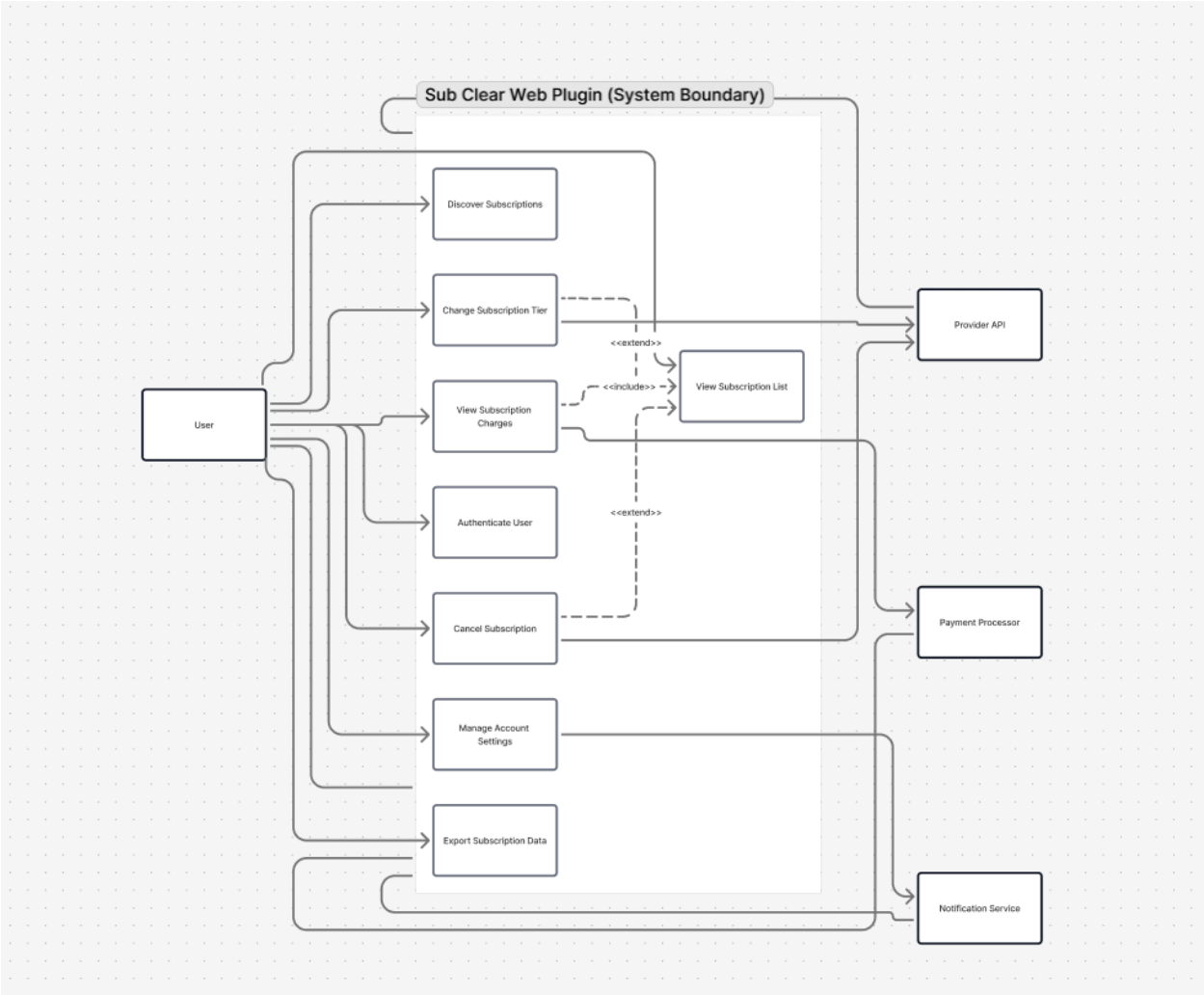
Problems:

Each team works together on use cases for your course project.

Your submission should include:

- A UML use case diagram
- The use case description table of at least one use case at the summary level
- The use case description tables of multiple use cases at the primary level
- The use case description tables of use cases at the sub-function level, if applicable

You should start each use case description table in a new page



Sub Clear Web Plugin: Comprehensive Use Case Analysis and Documentation

Introduction

The accelerated shift to subscription-based services has led to a proliferation of user accounts across entertainment, productivity, e-commerce, and content platforms. Users juggle dozens of recurring subscriptions, often losing track of charges, renewal dates, or even active memberships. This fragmented landscape introduces financial "leakage" and user frustration, stimulating demand for products that centralize the discovery, management, modification, and cancellation of digital subscriptions.

Sub Clear addresses this growing pain by offering a secure web plugin that empowers users to input their email and password, discover all subscriptions tied to that address—ranging from Netflix to SaaS tools, newsletters, and more—cancel or modify subscriptions, change plan tiers, and see all related charges in one place. Given the sensitivity, diversity, and complexity of this ecosystem, robust use case engineering, meticulous error handling, best-in-class UX, privacy compliance, and seamless integration with external services are mission-critical.

This report delivers a **deep analysis and documentation set for Sub Clear**, including:

- A standards-based UML use case diagram,
- A summary-level use case table,
- Detailed descriptions for the primary use cases,
- Considerations for error, compliance, and integration scenarios,
- References to contemporary best practices and real-world examples.

Throughout, references are made to leading sources on UML, subscription management, UX, error handling, and compliance, grounding the analysis in industry-standard methods and up-to-date requirements.

UML Use Case Diagram

(ABOVE)

Actors:

- **User**: Authenticates, manages subs, sees charges, exports, etc.
- **Subscription Service Provider (external API)**: Receives cancellation/change requests, returns status and details.
- **Payment Processor (external)**: Supplies transaction data if needed.

Relationships:

- User to Use Cases: association (user initiates actions),
- Includes: e.g., View Subscription Charges requires 'View Subscription List' completion,
- Extension points (not shown): error cases (invalid login, API timeout), compliance warnings.

Use Case Summary Table

Below is a summary table following industry-standard headers.

Use Case ID	Use Case Name	Primary Actor	Goal/Description	Related Actors	Trigger	Precondition	Postconditions
JC-1	Authenticate User	User	Securely log in and prove identity		User enters credentials	Valid email/password	Session/token created
JC-2	Discover Subscriptions	User	Identify and list all subscriptions tied to email	Subscription Svc, Payment API	After authentication	User authenticated	List displayed
JC-3	View Subscription List	User	Display found subscriptions by type/provider		After discovery	Discovery successful	Subscriptions shown
JC-4	View Subscription Charges	User	Show recurring/past charges and next billing date	Payment Processor, Svc Provider	User selects subscription	List retrieved	Charges displayed

JC-5	Cancel Subscriptio n	User	nitiate ancellation vith provider	Subscriptio n Svc Provider	User selects ancel	Subscription ctive	Cancel onfirmed/status
JC-6	Change Subscriptio n Tier	User	Upgrade/downgr ade plan tier	Subscriptio n Svc Provider	User selects hange	Service upports tier hange	Status updated
JC-7	Export Subscriptio n Data	User	Download list of ubs and charges		User requests xport	List loaded	Data file delivered
JC-8	Audit Trail Logging	System	Track hanges/actions or support and ompliance		On critical ctions	User uthenticated	Log created
JC-9	Manage Account Settings	User	Update email, assword, rivacy consents		User updates ettings	User uthenticated	Changes saved
JC-10	Error Handling & Support	System/Us er	Provide eedback and ecovery options or errors	Support ystem/exte nal APIs	On error or exception	Action failed	Message, escalation ath

Elaborative Summary

This table is deliberately detailed, supporting traceability and cross-functional collaboration among engineering, UX, security, and compliance teams. The inclusion of actors beyond the direct User (such as external system integrations) ensures that the real-world orchestration of use cases is reliably documented, reflecting actual dependencies and flows.

Detailed Use Case Descriptions for Primary Tasks

Each major use case is now described in detail using a comprehensive template, and then further analyzed, contextualized, and referenced with current best practices.

UC-1: Authenticate User

Summary: Enable users to log in securely via email/password, establishing a verified session for subsequent subscription management actions.

Primary Actor: User

Stakeholders:

- User: Needs privacy, security, and ease of use.
- Sub Clear: Must ensure robust authentication and prevent unauthorized access.
- Compliance: Must adhere to data protection (GDPR, CCPA, etc.).

Preconditions:

- User has an existing account (or registration flow is triggered).
- Secure HTTPS connection is established.

Triggers:

- User loads the Sub Clear interface and initiates login.

Main Flow:

1. User enters their email and password.
2. System validates email format and checks for an existing account.
3. System verifies password via secure backend.
4. If correct, create a session (token/cookie, JWT, etc.).
5. Proceed to subscription discovery.
6. Optionally, enforce multi-factor authentication.

Alternate Flows:

- A1: "Forgot Password" is triggered, system sends reset instructions.
- A2: Invalid password—system allows retry, shows a secure error message.
- A3: Account locked after repeated failures, triggers support workflow.

Postconditions:

- *Success*: Authenticated user with valid session.
- *Failure*: Inaccessible functions, possible error handling path.

Error Scenarios:

- Network timeouts, server unavailable, brute-force detected; all handled gracefully, with audit log.

Comments/UX Notes:

- UX must guide user with just-in-time feedback.
- Strong messaging around security/privacy.
- Consider adding support for social logins or SSO for enterprise/bundled subscriptions.

UC-2: Discover Subscriptions

Summary: Using user credentials, scan all possible third-party services (via APIs, mailbox parsing, payment data, or known patterns) to discover subscriptions linked to the user.

Primary Actor: User

Stakeholders:

- User: Wants accurate, complete discovery; minimum privacy risk.
- Sub Clear: Must balance thoroughness with least-privilege access.
- Service Providers: May require OAuth, token, or basic API authentication.

Preconditions:

- User is authenticated.
- Consent given for necessary access/scope—compliant with GDPR/CCPA.

Triggers:

- User requests discovery, typically auto-invoked post-login.

Main Flow:

1. Sub Clear requests permission to access user's email inbox and/or billing data (as needed).
2. System connects to mailbox using secure OAuth/IMAP/OAuth2, scanning for:
    - Subscription confirmation emails,
    - Renewal notifications,
    - "Welcome" or "Thanks for subscribing" patterns,
    - Billing statements.
3. System calls integrated APIs (where possible, e.g., Netflix, Spotify, newsletter platforms) to check subscriptions using user's credentials or OAuth.
4. System uses payment processors (Stripe, Plaid, Paypal, open banking APIs) to scan for recurring transactions—even for subscriptions without explicit email trails.
5. Aggregates and deduplicates results into an internal model.
6. User presented with a categorized list of discovered subscriptions, charges, billing cycles, and last activity.
7. User may approve integrations for advanced/locked services (e.g., Netflix API now private; must prompt for alternate confirmation—see limitations).

Alternate Flows:

- A1: API for a provider is unavailable (e.g., legacy Netflix): system requests user to upload PDF statement or manually confirm.
- A2: Privacy permission denied (user does not want inbox scanned), allow partial discovery via payments only.
- A3: Consent withdrawal—user can immediately revoke access and clean up data.

Postconditions:

- *Success*: Subscription list (with statuses) displayed to user.
- *Failure*: User alerted to subscriptions that couldn't be confirmed, with suggested next steps.

Error Scenarios:

- API rate limiting, OAuth failures, CAPTCHAs on email providers, or parsing errors; system should surface partial results and log for audit/compliance.

UX/Privacy Notes:

- Clearly explain (with permission screens or overlays) what rights are requested and why.
- Offer transparency—show which providers were checked, skipped, or couldn't be accessed.
- For mailbox parsing, describe and minimize data handling; implement least-privilege and strict data retention.

UC-3: View Subscription List

Summary: Display all discovered subscriptions in an organized user interface, supporting search, filtering, sorting, and navigation to further actions (view details, cancel, change plan, etc.).

Primary Actor: User

Stakeholders:

- User: Needs a clear summary to make informed decisions.
- Sub Clear: Must facilitate exploration and support next actions.

Preconditions:

- Discovery completed.
- User is authenticated and session is valid.

Triggers:

- Discovery completion or manual reload by user.

Main Flow:

1. System displays subscriptions grouped by category (e.g., Streaming, Productivity, Newsletters, Utilities).
2. For each subscription, show:
    - Service (name/logo, type),
    - Current tier or plan name,
    - Next billing date,

    - Last charge amount and date,
    - Status (active, canceled, expiring).
3. User can filter/search by provider, type, charge size, etc.
4. Clicking an item opens details—exposes more history, charge breakdown, and actions.

Alternate Flows:

- A1: No subscriptions discovered: show onboarding with advice or manual add option.
- A2: Some subscriptions could not be classified—group under "Other" with explanation.

Postconditions:

- User is enabled to inspect and act on each service.

UX/UI Best Practices:

- Sticky/fixed summary, progress bars, clear CTAs for "Cancel," "Change Plan," or "See Charges".
- Mobile-optimized layout and navigation; responsive controls.

UC-4: View Subscription Charges

Summary: Show recent and upcoming charges/billing cycles for each subscription. Visualize trends, highlight anomalies (sudden charge increases), and disclose all available cost information.

Primary Actor: User

Stakeholders:

- User: Desires transparency, budgeting support, fraud detection.
- Sub Clear: Must pull charge data from multiple sources; ensure accuracy.

Preconditions:

- Subscription is discovered and selected.
- User grants permission for payment data linking if required.

Triggers:

- User navigates to a subscription's details and requests charge info.

Main Flow:

1. System queries linked payment processor APIs (Stripe, Plaid, PayPal, or card provider APIs) for all transactions matching this subscription.
2. Cross-references with provider API data for upcoming bills or charges (if available).
3. Constructs a clean, time-ordered table/chart:
    - Date,

    - Amount,
    - Billing period (monthly, annual, etc.),
    - Status (paid, failed, pending).
4. Visual indicators highlight changes (e.g., charge increase, trial ending, missed payment).
5. For missing data (API doesn't provide, or user declined), system provides partial estimate and caveats.
6. User can download/export the charge history as CSV/Excel/PDF.

Alternate Flows:

- A1: Payment API requires additional security (two-factor, refresh token).
- A2: Payment API returns ambiguous charges—system groups and flags for manual review.

Postconditions:

- User is able to review, understand, and act on financial implications of each subscription.

Data/Privacy/Compliance:

- All payment/transaction data minimized and sandboxed; shown only with user consent.
- Audit trail entry created for each access/viewing transaction for fraud/legal defense.

UC-5: Cancel Subscription

Summary: Enable user-driven, seamless cancellation of individual subscriptions via provider APIs, direct authenticated web workflows, or—where required—manually.

Primary Actor: User

Stakeholders:

- User: Wants cancellations to be quick, transparent, and effective.
- Provider: May offer retention flow, alternate plan offers, pause/downgrade instead of cancel.
- Sub Clear: Must translate actions to diverse provider APIs and show clear status.

Preconditions:

- User is authenticated, discovery completed.
- Subscription is active/cancelable.
- (For some services) OAuth or direct login token for provider still valid.

Triggers:

- User selects "Cancel" from the subscription's UI card or details screen.

Main Flow ("Self-Service API or Integrated Cancellation"):

1. Sub Clear first checks if provider supports authenticated API cancellation (e.g., via OAuth, delegated API, or similar).
2. If yes:
    - System warns/reviews with the user on finality, possible lost benefits, and asks for confirmation.
    - On confirmation, system issues API cancellation request to the provider.
    - System receives status confirmation and updates UI instantly.
    - All steps logged.
3. If not supported, system may route user to:
    - In-context web widget (embedded provider cancel form, via secure iframe),
    - Direct link to provider's account page with pre-populated information (deep linking),
    - Step-by-step manual guidance, possibly automating steps (auto fill, copy/paste policy numbers, support email templates).

Best Practice Enhancement ("Retention/Win-back Offers"):

- Following modern SaaS best practices, at the point of cancellation, optionally:
    - Display personalized offers (discount, free month, "Are you sure?" nudge),
    - Collect reason for cancellation (survey; not required for completion).

Alternate Flows:

- A1: Cancellation fails (API error, insufficient permissions): system escalates to support, transparently explains to user.
- A2: Provider requires phone/email or manual confirmation: system provides guidance and templates.

Postconditions:

- *Success*: Subscription is marked as canceled both in the provider's and Sub Clear's system; user notified of refund or remaining service term.
- *Partial*: Request queued or handed off for manual processing.
- *Failure*: User notified and given follow-up options/support.

Legal/Compliance:

- All cancellation attempts result in audit-log entries with timestamps, request/response snapshots, and user notification.
- Must support "click-to-cancel" laws (especially US/California) and not introduce unnecessary friction or delays; prioritize user autonomy.
- Multilingual support for global users (per legal requirements and best practice).

UX Notes:

- UX should always make the irreversible nature of "cancel" clear and prevent accidental triggers (confirmation dialog).
- Where cancellation is not possible, document clearly for the user.

UC-6: Change Subscription Tier

Summary: Allow users to change the plan/tier (e.g., from Netflix Basic to Premium), including upgrades, downgrades, or switching to special rate plans, through direct provider API or guided manual workflow.

Primary Actor: User

Stakeholders:

- User: Seeks to optimize costs/features; expects real-time confirmation.
- Provider: Retains more users by making downgrade/upgrade easy.
- Sub Clear: Ensures updated next billing date/amount is reflected for transparency.

Preconditions:

- Discovery complete, subscription supports multiple tiers.
- Provider supports API/integration for tier modification or has web flow.

Triggers:

- User selects "Change Plan" or similar CTA in the UI.

Main Flow:

1. Sub Clear fetches and displays all available plans/tiers for the chosen provider via API or web scraper (if API unavailable).
2. User reviews features, prices, and select a preferred plan.
3. Sub Clear prompts for confirmation, stating new price, features, and when billing switch will occur.
4. On user approval, executes the change via API (POST/PUT) or automates the web process.
5. Updates charge view, next renewal, and plan status in UI.

Alternate Flows:

- A1: Provider only supports manual upgrade (no API)—Sub Clear displays deep link, preps user with required info to complete on provider’s site, and requests confirmation when done.

Postconditions:

- *Success*: New tier effective, confirmation shown.
- *Partial*: User asked to confirm later (e.g., provider needs manual approval).
- *Failure*: User notified; support suggestion provided.

Compliance / Logging:

- All tier change requests are logged for user support and dispute resolution.
- Email or in-app notifications confirm change.

UX Notes / Best Practices:

- Show clear tier comparison (price/features/benefits), including prorated implications.
- Allow user to preview future billing impacts before confirming.
- Alert if provider has any non-obvious rules (e.g., cannot downgrade in mid-cycle, must finish trial first).

UC-7: Export Subscription Data

Summary: Allow users to download or transfer their subscription and charge data for offline record-keeping or portability.

Primary Actor: User

Stakeholders:

- User: Desires backup, analysis, or regulatory access to their own data.
- Sub Clear: Must support GDPR/CCPA "right to data portability".

Preconditions:

- Discovery complete, user authenticated.

Triggers:

- User initiates export from UI.

Main Flow:

1. System assembles current subscription/charge dataset in standard machine-readable format (CSV, JSON, PDF).
2. Optionally, user can select range/filter (e.g., only active, or only certain services).
3. System packages data and serves for download, with timestamp in filename.
4. Export action is logged for audit trail.

Alternate Flows:

- A1: User requests export for a regulatory request (DSAR)—tag for compliance logging.

Postconditions:

- Data delivered securely; deletion/portability demands can be fulfilled.

Compliance:

- Export process is frictionless; no hidden lock-in.

UC-8: Audit Trail / Logging

Summary: Every critical user/system action is tracked—from authentication to changes/cancellations—for compliance, security, support, and forensic investigation.

Primary Actor: System

Stakeholders:

- User: Protected in case of disputes/errors; ensures accountability.
- Sub Clear: Fulfills legal compliance, supports operational health.

Preconditions:

- User authenticated or system-level event occurs.

Triggers:

- On any state-changing API call, error, data export, or support contact.

Main Flow:

1. System creates structured audit log record (timestamp, user, action, affected resource, status/result, context/metadata).
2. Audit logs are stored in append-only (WORM) storage or service.
3. Access is protected; visible to support or on user request.
4. System supports querying logs for compliance audit or post-mortem.

Postconditions:

- Traceability for every major user-system action.

Compliance:

- Conforms to SOX, GDPR, HIPAA, and modern best practices.

UC-9: Manage Account Settings

Summary: Let users manage their profile, password, notification settings, privacy consents, and delete account/data (e.g., "right to be forgotten").

Primary Actor: User

Stakeholders:

- User: Control over their account and information.
- Sub Clear: Fulfills privacy/consent requirements.

Preconditions:

- User is authenticated.

Main Flow:

1. User accesses "Settings" or "Account" panel.
2. Can update contact info, change password, manage alerts, and opt-in/out of marketing.
3. User can request account deletion, triggering irreversible removal and notification.
4. Changes logged with timestamp for traceability.

UX/Compliance:

- UI flows are clear, must request explicit confirmation for destructive actions.
- All privacy options documented; logs support compliance needs.

UC-10: Error Handling & Support

Summary: System detects and responds to any error, providing clear messages and recovery options, and logging for support/escalation.

Primary Actor: System/User

Stakeholders:

- User: Needs confusion-free guidance and assurance.
- Sub Clear: Support/DevOps, for bug tracking and resolution.

Preconditions:

- Error, exception, or expected sequence break (e.g., API fails).

Main Flow:

1. System detects error (code, exception, or timeout).
2. Parses error, matches to human-readable explanation.
3. Displays appropriate message in UI, suggesting recovery or next step.
4. Severe/recurring errors prompt escalation—offer direct support ticket or chat/email.
5. All error events are logged with context.

Best Practices:

- Never show raw stack traces to user, but make detailed logs available for dev/support.
- Implement retry, fallback, and user education (FAQ links).
- UX must reinforce trust; simple language, offer empathy.
- If problem is provider-side or systemic, communicate proactively (banner, email).

Additional Use Cases and Complex Scenarios

Integration with External Subscription Services

Given the vast diversity of providers (some with APIs, many without), Sub Clear must architect for:

- **Direct API Integration**: For SaaS/streaming services with documented APIs—use OAuth where possible, refresh tokens as needed.
- **Screen Scraping/Headless Browsing**: As fallback, employ automation to mimic browser actions for cancellation/change, with user consent.
- **Email/Parse-Driven Discovery**: Rely on robust NLP/parsing for providers lacking integration.

Challenges:

- Non-uniform cancellation/modification flows,
- Breaking changes in provider UIs,
- Legal/contractual limitations for API screening.

References:

- Modern subscription billing systems employ similar patterns, often relying on a mix of API and scraping depending on the provider.

Authorization and Role-Based Access (RBAC)

While the primary actor is the end-user, the platform should anticipate:

- Multiple users per email (family accounts),
- Admin/support staff requiring restricted access for troubleshooting (never full data access),
- Integration partners or bots for automation (rate-limited, restricted scopes).

Data Privacy, Consent, and Compliance

Full compliance with GDPR, CCPA, and similar is mandatory:

- Consent gates—every data access must have clear, granular opt-in, and *easy* opt-out.
- Minimal retention—ephemeral storage of credentials where possible.
- Export and deletion workflows for users (DSAR, right to be forgotten),
- Data storage in secure, regionally-appropriate infrastructure; thorough documentation.

Logging and Audit Trail Best Practices

Audit logs must cover:

- Login events (successful/failed), account changes,
- All subscription cancellations/changes/logins,
- Export/downloading data,
- API integration errors or outages.

Use WORM storage where possible, include context (IP, session, correlated events), and make audit histories available as needed for legal, user, or regulatory review.

User Interface and UX Patterns

- Use established best practices for subscription flows: dashboard summaries, clear callouts, snackbars, and modal confirmations for destructive actions,
- Progress indicators for lengthy discovery/cancellation,
- Accessibility (ARIA compliance, keyboard navigation), mobile-first layouts,
- Personalization and transparency—show onboarding, tips, and context-aware help where needed.

Error Handling and Exception Scenarios

Exemplary error handling includes:

- Detection of provider/API changes,
- Recovery (retry, alternate path suggestion),
- Proactive alerting (if an integration is known to be broken, notify user on login),
- Logging for diagnosis without exposing sensitive info,
- Incomplete or partial success should be clearly reported but not block other actions（support incremental success）.

Real-World References and Industry Benchmarks

- **Netflix, Stripe, Twilio, Slack, and similar**: Help center designs, user-driven flows, robust documentation and support for discovery/troubleshooting, and well-designed cancellation/retention mechanisms.
- **Churnkey, FunnelFox, ProsperStack**: Best-practice cancellation/retention flows, illustrating powerful retention nudges and compliance with new "click to cancel" legal requirements.
- **Stripe Billing, BluLogix, BillingPlatform, etc.**: Subscription and payment data management patterns; robust error handling; user empowerment for tier management.
- **Modern privacy platforms (SubTracker, Microsoft Purview, GDPR tools)**: Data export, deletion, explicit consent, lawful data use notifications, and response to regulatory demands.

Conclusion

Sub Clear, as a subscription management web plugin, must map to the highest standards of usability, privacy, reliability, and legal compliance. By formalizing the above use cases—grounded in exhaustive references and industry current practice—and supporting documentation, the product provides a single source of functional truth for developers, UX designers, compliance officers, security teams, and support staff.

Key takeaways:

- Comprehensive use case documentation ensures all core and edge flows are understood, supported, and future-proofed,
- Modern UI and audit systems offer user empowerment while delivering transparency and trust,
- Regulatory and privacy compliance is designed in, not bolted on,
- A culture of logging, auditing, and proactive error recovery ensures operational resilience and user satisfaction.

This approach not only accelerates development velocity and lowers risk but also positions Sub Clear as a best-in-class solution in the growing subscription management space.