Detailed Plan for Slide 4 (Plan and Strategy) and Slide 5 (Action and Execution)

Slide 4: Plan and Strategy

1. Audit Existing Documentation

How to Do It:

- Collect all documentation (APIs, SDKs, onboarding guides, FAQs).
- Create an inventory:
  - Tools: **Google Sheets** or **Airtable** to track documentation by title, type, owner, and last updated date.
  - Categories: Identify document types (e.g., reference vs. tutorial) and their target audience (e.g., developers vs. clients).
- Evaluate for:
  - Accuracy (Is the content up to date with the current platform?)
  - Clarity (Does it explain concepts or assume prior knowledge?)
  - Completeness (Are steps or examples missing?)

Questions for the Team:

- What documents do you use most often, and which are hardest to understand?
- Have you encountered errors in the documentation while working with clients?
- Are there recurring questions that documentation doesn't answer?

---

2. Gather Stakeholder Feedback

How to Do It:

- Schedule meetings with key stakeholders:
  - Developers: To understand their pain points and clarify technical details.
  - Customer Support: To learn about recurring issues raised in tickets.
  - Sales/Onboarding Teams: To identify what clients struggle with during integration.
- Conduct surveys for broader input:
  - Tools: **Typeform** or **Google Forms**.
  - Questions:
    - What do you find most frustrating about our documentation?
    - How often do you use documentation, and what is your preferred format?
    - Are there features or workflows you find particularly hard to implement?

3. Prioritize Updates

How to Do It:

- Use a **RICE Framework** to evaluate each document update:
  - **Reach**: How many users are impacted by this document?
  - **Impact**: How much will fixing it improve the user experience?
  - **Confidence**: How certain are we about its importance?
  - **Effort**: How much time and resources are required to fix it?
- Focus on high-impact, low-effort wins first (e.g., common APIs with outdated examples).

---

4. Plan Tools and Workflow

Key Tools:

- **Swagger/OpenAPI**: To update API documentation with live testing capabilities.
- **Confluence**: For collaborative writing and feedback gathering.
- **Markdown Editors**: Tools like Visual Studio Code or Obsidian for writing developer-focused content.
- **Draw.io** or **Lucidchart**: For creating diagrams and architecture visuals.

Workflow:

1. Create tickets in a project management tool like **Jira** for tracking updates.
2. Assign owners for each section of the documentation.
3. Schedule bi-weekly check-ins to ensure progress and gather feedback.

Slide 5: Action and Execution

1. Audit and Organization

- Create a **Content Map**:
  - List all documentation and group them by logical categories (e.g., onboarding, troubleshooting, advanced use cases).
  - Example tools: Use **Miro** or **Notion** for a visual map.
- Identify missing or redundant content:
  - Cross-reference against feedback from stakeholders.
  - Highlight documents that need consolidation or removal.

---

2. Standardization

- **Develop a Style Guide**:
  - Create sections for tone, structure, and technical conventions (e.g., consistent use of parameters, formatting for code snippets).
  - Use tools like **Grammarly** and **Hemingway App** to maintain clarity and tone.
- **Training**: Conduct workshops with internal teams to align on the new standards.

Key Elements to Standardize:

- API Examples: Every example includes input/output, context, and potential errors.
- Terminology: Avoid jargon and ensure terms are used consistently.
- Visuals: Define rules for diagrams (e.g., color schemes, naming conventions).

---

3. Content Updates

- **API Documentation**:
  - Use **Swagger/OpenAPI** to auto-generate interactive documentation.
  - Add use cases for each endpoint, showing how it integrates into a real-world OTT scenario.
- **Tutorials and Onboarding**:
  - Rewrite in step-by-step format with screenshots and diagrams.
  - Include a "Common Pitfalls" section to address frequently asked questions.
  - Example: If onboarding involves creating a custom player, include a fully functional example with reusable code snippets.

Collaborate With Developers:

- Schedule review cycles where developers validate technical accuracy.
- Use version control tools like **GitHub** to track changes.

---

4. Navigation Overhaul

- **Design New Information Architecture**:
  - Use card-sorting techniques to group documents logically.
  - Tools: **Optimal Workshop** to test navigation concepts with end-users.
- **Search Functionality**:
  - Implement a faceted search engine like **Algolia** or **ElasticSearch**.
  - Example: Allow filtering by content type (e.g., FAQ, API reference) and difficulty level.

---

5. Feedback Loop

- Add a **Feedback Widget**:
  - Tools like **Hotjar** or custom forms to capture real-time developer feedback.
  - Example: "Was this page helpful?" with optional comments for improvement.
- **Analytics**:
  - Use tools like **Google Analytics** or **Amplitude** to track page views, bounce rates, and user journeys.
  - Identify underperforming content and prioritize updates.