Google Summer of Code 2025 - Contributor Proposal : Drupal

Personal Information

Name: Ritesh Kumar Singh

Email: ritesh.davv@gmail.com

School/University: Institute of Engineering and Technology (IET) DAVV, Indore, M.P. India

Graduation Date: 2027 (currently in second year)

Major/Focus: Bachelor of Engineering (B.E.) in Electronics and Telecommunication

Which country do you live in?: India

What timezone is that in?: GMT +5:30 (UTC +5:30)

Link to your drupal.org profile: https://www.drupal.org/u/riteshdavv

What is your Drupal Slack ID?: riteshsingh

Preferred time of day for virtual/video interview: UTC 5:00 PM (IST ~10:30 or 11 PM)

What is your Open Source Experience so far?:

I'm new to open-source but highly enthusiastic about contributing. I've been actively exploring GitHub projects, studying contribution guidelines, and building my understanding of collaborative development. Through this GSoC project, I'm excited to make my first impactful open-source contribution and grow within the community.

GitHub: github.com/riteshdavv

LinkedIn: linkedin.com/in/riteshdavv

Portfolio: riteshsingh.vercel.app

GSOC Information

Have you participated in Google Summer of Code previously? Please describe your participation in Google Summer of Code.

No, this is my first time applying to GSoC. I'm very enthusiastic about this opportunity to dive deeper into open-source development through the Drupal community.

Are you applying to any other organizations this year? If so, please explain.

Yes, I have also submitted a proposal to another organization in a different domain as a backup. However, the Appwrite-Drupal integration excites me the most and aligns closely with my interests and experience.

How many hours will you devote to your GSoC project each week? What are your other summer plans?

I can devote approximately **40 hours per week** during the GSoC coding period. My primary focus this summer is to work on the GSoC project and contribute meaningfully to the Drupal ecosystem.

Have you registered an account at Drupal.org and joined groups.drupal.org/google-summer-code? Link to your account please!

Yes. Here is the link to my account: https://www.drupal.org/u/riteshdavv

Have you ever worked with Git?

Yes, I use Git regularly for all my personal, academic, and collaborative projects. I am proficient in managing branches, resolving merge conflicts, and maintaining clean commit histories.

Question based on knowledge of git:

You just committed some code (and not pushed yet), but you realized there is a typo in the commit message. How would you change it (please explain each step of the solution)?

I would run the following command to amend the last commit message:

```
git commit --amend
```

This will open my configured text editor to change the message. After saving and closing, the new message will replace the previous one. Since the commit is not pushed, this is safe to do.

Have you ever contributed code to Drupal? If you have, tell us about it.

Not yet, but I have installed and explored Drupal on my local system using DDEV. I'm eager to begin contributing through this GSoC project and beyond.

Do you plan to continue contributing to the Drupal project after GSoC is finished?

Absolutely. I believe Drupal is a powerful platform with a welcoming community, and I plan to continue contributing to modules, patches, and issue queues after GSoC.

Have you ever built a Drupal site or helped on a Drupal project?

I have locally installed and experimented with Drupal using DDEV. While I haven't built a full-fledged production site yet, I have explored Drupal modules, backend structure, and content workflows.

If you have not previously contributed to any open source projects, can you provide example code from school?

Yes, I can provide code samples that demonstrate my backend integration work, including authentication workflows, RESTful services, and file handling. These can be found in my GitHub repositories.

1. Question based on given 'almighty_function'

```
function almighty_function($x, $y, $z) {
  if ($y != $z && $x == $y && $x == $z) {
    return "Success!";
  }
  return "FAIL!";
}
```

Set of values for Success:

There is **no valid set of values** that can satisfy the condition

- `($y != $z && $x == $y && $x == $z)` simultaneously,
- because `$x == $y` and `$x == $z` implies `$y == $z`,
- which contradicts `$y != $z`.

Therefore, it is **impossible** for the function to return `"Success!"`.

---

2. Describe 5 conceptual layers in a Drupal system.

1. **Theme Layer:** Handles the visual appearance of the site using templates, CSS, and JS.
2. **Presentation Layer (Render API):** Converts data structures into renderable HTML.

3. **Business Logic Layer (Modules):** Contains custom and contrib modules that process content, user permissions, and logic.
4. **Data Access Layer (Database API):** Manages database operations abstractly through APIs.
5. **Configuration Layer:** Stores and manages site settings, module configurations, and environment-specific options.

Appwrite Integration Module for Drupal

Project Information:

Which project idea sparks your interest and why?

The project idea to develop an Appwrite integration module for Drupal deeply interests me because it bridges modern backend technologies with a mature, widely-used CMS. Appwrite is a rising open-source BaaS platform offering APIs for authentication, storage, and databases, and Drupal is one of the most trusted CMS frameworks in the enterprise world. Integrating the two not only opens up a new backend architecture possibility for Drupal developers, but also promotes flexibility, scalability, and developer experience in content management systems.

With my background in backend development, REST APIs, OAuth, and open-source contribution, I'm excited to take this opportunity to build a module from scratch that connects two powerful ecosystems. I'm particularly drawn to the technical challenge of creating a seamless and configurable integration layer, while making it accessible and developer-friendly for the Drupal community.

Detailed Implementation Plan (with Weekly Timeline)

This plan outlines the implementation of the **Appwrite Integration Module for Drupal** across the full GSoC period. It includes weekly milestones, technologies involved, communication rhythm with mentors, and when to expect tangible deliverables.

Community Bonding Period (May 20 – June 16)

Goals:

- Finalize feature scope and expected MVP
- Align with mentors on technical stack and architecture decisions
- Set up work environment and dev tools

Tasks:

- Set up Appwrite and Drupal local environments
- Deep dive into Appwrite's authentication, storage, and document database APIs
- Review existing Drupal modules using OAuth and external API integration
- Discuss structure of admin configuration UI

Communication:

- Slack: Daily active
- Weekly 1:1 Zoom/Meet call with mentors
- Shared Notion/Trello board setup for issue tracking and status updates

Deliverables by June 16:

- Finalized Architecture Plan
- Development environment ready
- Trello board and GitHub repo live

Phase 1: Authentication + Admin Interface (June 17 – July 22)

Week 1 (June 17 – 23): Module Bootstrap & Configuration Framework

- Scaffold Drupal module using Drupal Console
- Define folder structure: `src`, `config`, `services`, `forms`, etc.
- Begin implementing configuration form (Appwrite Project ID, Endpoint, API Keys)
- Build Appwrite service class (singleton pattern) to encapsulate API logic

Goal: Have a working skeleton module with configuration UI

Week 2 (June 24 – 30): OAuth Integration (Google & GitHub)

- Implement Drupal service for handling Appwrite OAuth sessions

- Use Appwrite PHP SDK to trigger provider-based login flows
- Store user sessions securely and map Appwrite users with Drupal user roles
- Handle token storage, refresh, and error states

Goal: Functional login with Google/GitHub OAuth into Drupal via Appwrite

---

Week 3 (July 1 – 7): User Sync + Registration Flow

- Hook into `user_login`, `user_register` events in Drupal
- Ensure synced login state between Appwrite and Drupal users
- Add fail-safes for OAuth mismatch cases
- Create UI notification system for error/debug messages

Goal: Bi-directional user login flow fully working

Week 4 (July 8 – 14): Admin UI Polish + OAuth Extensibility

- Add support for additional providers (Apple, Facebook) via config
- Enhance configuration form (Form API validation, instructions, save state)
- Add debug/test mode for validating Appwrite connectivity
- Start documenting usage of authentication flow

Goal: Fully polished, extensible authentication + admin config interface

---

Week 5 (July 15 – 22): Media Storage System with Appwrite Bucket API

- Research Appwrite's storage and bucket API system
- Build integration service in Drupal for uploading files to Appwrite
- Modify Drupal's media file handling to point to Appwrite storage buckets
- Create fallback for using local files when API is unavailable

Goal: Functional media file upload + retrieval using Appwrite Storage

---

Midterm Deliverables (by July 22):

- Working OAuth integration (min 2 providers)
- User sync system between Drupal and Appwrite
- Appwrite-powered media file upload & config panel
- Documentation for setup and usage
- Code review and unit test for core flows

Phase 2: Document Database, Testing & Packaging (July 23 – August 19)

Week 6 (July 23 – 29): Document Database Integration Begins

- Explore Drupal content structure: nodes, fields, metadata
- Implement Appwrite document CRUD APIs inside a Drupal service
- Allow custom content types to sync with Appwrite documents
- Maintain a mapping system between Node ID ↔ Appwrite Document ID

Goal: One-way sync (Drupal → Appwrite) for selected content types

---

Week 7 (July 30 – Aug 4): Full Content Sync + Retrieval

- Enable document retrieval (Appwrite → Drupal) for custom blocks/pages
- Add permissions and access control (Appwrite rules + Drupal user roles)
- Add a toggle in config panel to enable/disable sync for specific types
- Integrate error logging for failed syncs

Goal: Fully functional content sync engine with control switches

---

Week 8 (Aug 5 – 11): Final Testing, CI Setup, and UX Improvements

- Write unit tests and test Appwrite endpoints via stubs
- Test module in different environments (Lando, DDEV, Docker)
- Add Drupal permissions page support for admin roles

- Polish UI of configuration panel and make mobile/tablet responsive

Goal: Stability & UX improvement across all components

Week 9 (Aug 12 – 18): Final Documentation + Submission

- Write full technical documentation (developer + end user)
- Create README with examples and setup guide
- Record video tutorial and walkthrough of all features
- Push final code to GitHub with appropriate LICENSE

Goal: Module fully ready for production + contribution to Drupal community

---

Summary of Expected Deliverables

Timeline	**Deliverable**	Description
Week 1–2	**Module structure + OAuth integration base**	Set up the foundational structure of the Drupal module including folder layout, service architecture, and configuration interface. Initial OAuth integration with Appwrite begins, including support for at least one provider like Google. The goal is to have a solid base from which to build out all further integrations.
Week 3–4	**Authentication + admin UI completed**	Fully functional authentication flow using Appwrite OAuth, with user session mapping between Appwrite and Drupal. Also includes a complete and intuitive admin configuration panel within Drupal where site administrators can enter Appwrite credentials, toggle settings, and test integrations.

Week 5	**Appwrite storage integration**	Enables Drupal to offload media file management to Appwrite's object storage. Media uploads and retrievals will use Appwrite's Bucket API, helping Drupal sites reduce reliance on local or traditional cloud storage. This includes fallback logic and admin toggles.
Midterm Evaluation	**OAuth + Storage + Admin UI**	By this point, the module should support full Appwrite-based authentication, media file storage, and admin configuration. It will be well-documented, tested in at least one deployment environment, and ready for mentor evaluation with a demoable version.
Week 6–7	**Document database sync**	Appwrite’s document database will be integrated to store and retrieve structured Drupal content. Content types (nodes) can be selectively synced to Appwrite, and mapped via unique IDs. Includes logic for CRUD operations and error handling on sync failures.
Week 8	**Testing + permissions + CI setup**	This phase focuses on module hardening. Includes writing automated unit tests, integrating continuous integration (CI) tools (e.g., GitHub Actions), and ensuring permission systems in both Drupal and Appwrite align correctly to ensure secure usage.
Week 9	**Docs + Code finalization + Submission**	Final polishing and packaging of the module. Includes full developer and user documentation, recorded walkthrough/demo, README with setup instructions, and public GitHub repository push. The module will be ready for production use or submission to the Drupal community as a contributed module.

This timeline is built with buffer time in mind for testing, feedback loops, and unforeseen challenges. If new Appwrite features or community feedback arise mid-project, there is room to adapt. I’m confident this plan gives a clear direction for turning the proposal into a real, usable module.

Mentor Communication

To maintain transparency, clarity, and a steady feedback loop throughout the project, I will adopt a structured multi-channel communication approach that blends synchronous and asynchronous methods for maximum efficiency.

- **Primary Communication:** Weekly 1:1 sync-ups via Google Meet or Zoom every Friday to review progress, address blockers, and align on goals.
- **Asynchronous Collaboration:** Slack will serve as the main channel for daily updates, clarifications, and mentor interaction across time zones.
- **Progress Tracking:** A shared GitHub Project Board will be used for PRs, milestones, and issue tracking, supported by Trello for sprint planning.
- **Bi-weekly Reports:** Detailed summaries and progress check-ins will be shared every two weeks via email in markdown format, ensuring traceability and documentation.

This system promotes consistent mentor engagement while offering flexibility to adapt as the project evolves.

Tools for Project & Time Management

Tool	Purpose	Description
Trello	**Task Planning and Prioritization**	Trello will be used to organize tasks into weekly sprints using boards and cards. Each task will have deadlines, checklists, and labels for progress tracking (e.g., To-Do, In Progress, Done). It ensures transparency and helps both the contributor and mentors track development at a glance.
GitHub	**Version Control + Issue Tracker**	All source code will be managed through GitHub using feature branches and pull requests for review. Issues will be created for bugs, improvements, and feature tracking. Milestones will align with weekly deliverables and evaluations, supporting efficient collaborative development.

Notion	**Architecture + Notes + Roadmaps**	Notion will serve as the internal wiki for the project. It will hold architecture diagrams, design decisions, setup guides, integration plans, and brainstormed ideas. Notion also helps maintain clarity around scope and future integration considerations.
Google Docs	**Progress Reports + Drafts**	Weekly and bi-weekly progress updates will be shared with mentors through Google Docs. Additionally, technical documentation, feature outlines, and written drafts (e.g., final report, midterm feedback) will be collaboratively edited here.
Google Meet	**Weekly sync with mentors**	Regular 1:1 or 1:2 video meetings with mentors via Google Meet will be scheduled weekly (or bi-weekly if needed). These sessions will be used to discuss blockers, feedback, upcoming goals, and integration decisions to maintain alignment throughout the project.

Detailed Description (only if this is a new idea, otherwise if there is a description online, please provide the URL):

This is a new idea. The project aims to develop a Drupal module that integrates core Appwrite services (OAuth-based authentication, object storage, and document database) directly into the Drupal CMS via a configuration-friendly interface. Future extensibility will also be considered for functions like serverless tasks and messaging.

Appwrite is an open-source Backend-as-a-Service (BaaS) that offers a comprehensive suite of tools—ranging from authentication and storage to databases and serverless functions—through easy-to-use APIs. This project aims to develop a Drupal module that integrates Appwrite's services, giving Drupal developers a robust alternative for handling key backend functionalities. The module will primarily focus on connecting Drupal with Appwrite's authentication system, object storage, and document database, while also laying the groundwork for future integrations like serverless functions, messaging, and real-time event handling.

Many Drupal sites currently rely on local media storage or traditional cloud services, but with Appwrite's flexible architecture, we can streamline media management and content handling. The solution will include a dedicated configuration interface within Drupal, making it simple for site administrators to set up and manage Appwrite settings.

Who are your mentors:

- **Abhinav Jha** (abhinavnarayan2002@gmail.com)
- **Ujjval Kumar** (ujjvalkumar113@gmail.com)

Which aspect of the project idea do you see as the most difficult?

The most challenging part of the project will likely be architecting a seamless, secure, and future-proof integration between Drupal and Appwrite—two systems with different design philosophies. Drupal's module and service-based architecture needs to be harmonized with Appwrite's modern API-first approach, particularly around authentication. Special care must be taken to ensure user identity mapping, token handling, and long-term extensibility are reliable and secure.

Challenges include:

- Designing a modular integration that adheres to Drupal's architecture (hooks, services, DI).
- Aligning Appwrite's OAuth flows with Drupal's user entities and session management.
- Handling token lifecycles (refresh, revocation, expiry) securely.
- Adapting to evolving Appwrite APIs without breaking the Drupal module.
- Ensuring scalability and extensibility for future integrations (e.g., serverless functions).

Which aspect of the project idea do you see as the easiest?

The most straightforward aspect of the project is implementing the configuration interface using Drupal's Form API and integrating Appwrite's object storage. Both platforms have strong documentation and SDK support, allowing for quick development and reliable testing.

Reasons this part is easier:

- Drupal's Form API simplifies creation of structured and validated admin interfaces.
- Appwrite's object storage APIs are clearly documented with SDK support in multiple languages.
- This task is mostly UI-driven and doesn't involve deeply complex backend logic.

- It will allow early hands-on exposure to both systems, reinforcing understanding for later stages.

Which portion of the project idea will you start with?

To lay a strong foundation, I will begin with integrating the OAuth-based authentication workflow. This step is critical to ensuring secure user authentication, which is a prerequisite for most other interactions between Drupal and Appwrite.

Reasons to start with OAuth:

- Authentication is foundational for all protected interactions (storage, documents, etc.).
- Early OAuth success helps test and validate the architecture and integration strategy.
- Enables testing of user-specific access control and session behavior in Drupal.
- Ensures secure, scalable access before building dependent modules.

How will you deal with project, task, and time management?

I'll follow an agile, milestone-driven approach with clear weekly targets, active mentor feedback loops, and transparent progress tracking. I plan to use a combination of tools tailored for collaboration, documentation, and sprint tracking to keep the project organized.

Tools and strategies:

- **Trello** – For weekly sprint planning, task assignment, and visual tracking (To-Do, In Progress, Done).
- **GitHub** – For version control, collaborative PR reviews, and issue management.
- **Notion** – To maintain architectural diagrams, brainstorming notes, and roadmap documentation.
- **Google Docs** – For writing technical documentation, sharing bi-weekly progress updates, and drafting content.
- **Slack & Google Meet** – For weekly sync-ups with mentors and instant communication.

Regular syncs, documentation, and continuous review will ensure the project stays on track and is delivered with high quality.

Biography/About Me

I’m a backend developer, open-source contributor, and full-stack tinkerer who thrives on building modular, API-driven, and impactful solutions. My background spans PHP, Symfony, RESTful APIs, OAuth flows, and integrating modern BaaS platforms like Firebase and Supabase—experience that directly aligns with the goals of this project. Over time, I’ve built and contributed to Drupal-based systems and strongly value code maintainability, extensibility, and clarity in documentation.

Beyond traditional development, I also enjoy pushing the envelope of creativity and humor in tech. I’ve developed AI-driven web platforms that blend frontend finesse with backend intelligence, using frameworks like Next.js and FastAPI, along with tools like HuggingFace and D3.js. These projects reflect my love for building technically challenging, delightfully weird, and data-driven tools that are both useful and engaging.

I see this GSoC project not just as a technical milestone but as an opportunity to empower the Drupal ecosystem with modern, scalable backend capabilities powered by Appwrite. I’m eager to bring this vision to life with community collaboration and open-source spirit.

Key Highlights

- **Backend-Focused Developer**: Experienced in PHP, Symfony, and OAuth integrations; comfortable building modular systems that align with Drupal’s service architecture.

- **Open Source Contributor**: Active on GitHub with experience in collaborative, documented, and community-first development.

- **Creative Tech Projects**:

    - **BuggedIRL** — an AI-powered bug-to-meme generator that turns frustrating errors into shareable comic relief. Link to the deployed website

        - **Tech Stack:** Next.js, Python (FastAPI), HuggingFace API, Meme Generation API (ImgFlip)

        - **Workflow:** Users input a bug or error → AI analyzes the issue and writes a funny summary → a meme is generated based on the AI text → users can share the meme.

        - Link of the Repository: github.com/riteshdavv/buggedirl

- **GitShamed** — an AI-powered “Git Roast” platform where users connect their GitHub repositories and receive humorous roast-style reviews of their code.

  Link to the deployed website

  - **Tech Stack:** Next.js, Python (FastAPI), HuggingFace API, GitHub API
  - **Workflow:** Users connect a repo → backend fetches files & analyzes code smells → AI generates sarcastic roasts → the UI displays hilarious feedback with visuals.
  - Link of the Repository: github.com/riteshdavv/gitshamed

- **GitWrecked** — a GitHub analytics dashboard that transforms your contributions into a gamified visual report. Link to the deployed website

  - **Tech Stack:** Next.js, GitHub API, Chart.js, D3.js
  - **Features:** Visual heatmaps of activity, pull request progress, issue tracking stats, and a motivational dashboard to showcase coding streaks and milestones.

  Link of the Repository: github.com/riteshdavv/gitwrecked

- **Tech Storyteller**: I enjoy building projects that don't just solve problems but also bring personality, fun, and motivation into developer workflows.
- **Excited for GSoC**: Motivated to contribute a robust Appwrite integration to Drupal that blends security, usability, and long-term maintainability.

Add a one minute video link here:

https://drive.google.com/drive/folders/1I3JPP1pldTvaEiGXE0g7vqdgWylVgEGz?usp=sharing

An online CV/Resume if you have one:

https://riteshsingh.vercel.app

Benefits to the Community

Drupal is a powerful and extensible content management system, but it often lacks native support for modern backend services without custom development. This project bridges that gap by **integrating Appwrite—a robust open-source Backend-as-a-Service—directly**

into Drupal, empowering developers to build more scalable, secure, and flexible applications with ease.

The proposed module will provide **seamless integration of OAuth-based authentication, object storage, and document database** capabilities into the Drupal ecosystem. It will eliminate the need for fragmented workarounds and enable **developers, site administrators, and organizations to adopt modern backend infrastructure** without compromising on Drupal's strengths.

With experience building **scalable, API-driven full-stack platforms** using technologies like **Next.js, MongoDB, Express, and REST APIs**, I bring a strong foundation in architecting both frontend and backend systems. This project allows me to apply that expertise toward creating a **developer-friendly, future-ready Drupal module** that enhances usability and accelerates adoption of Appwrite within the Drupal community.

Key Benefits:

- **OAuth Integration Simplified** – Easily connect social login providers like Google, GitHub, and Apple via Appwrite's secure authentication system.
- **Cloud-Native Media Handling** – Replace traditional storage methods with **Appwrite's scalable object storage**, streamlining media management.
- **Modular & Extensible Design** – Lays the groundwork for **future integrations** such as serverless functions, real-time messaging, and event handling.
- **Admin-Friendly Configuration UI** – A thoughtfully designed UI within Drupal to **manage Appwrite settings effortlessly**, reducing setup complexity.
- **Comprehensive Documentation** – Detailed guides and technical docs will ensure **long-term usability, community contributions, and maintenance**.

By bringing Appwrite's modern backend capabilities to Drupal, this project will **future-proof content management workflows**, support **API-first application development**, and open up **new possibilities for the Drupal ecosystem—**from startups and nonprofits to large-scale enterprises.

System Architecture

The Appwrite Integration Module is designed to bring scalable, modern backend capabilities to Drupal by bridging its content management ecosystem with Appwrite's powerful BaaS stack. The architecture focuses on modularity, extensibility, and clean abstractions for authentication, storage, and content management workflows.

- **OAuth Authentication Bridge**

  Leverages Appwrite's OAuth2 providers (Google, GitHub, Apple, etc.) to authenticate Drupal users. The system **maps Appwrite sessions** to Drupal's user model through a secure token management layer, enabling smooth and secure logins.

  - Token refresh and revocation handling
  - User session mapping between Appwrite and Drupal
  - Configurable provider list from the admin interface

- **Object Storage Integration**

  Integrates **Appwrite's object storage API** to enable efficient media handling in Drupal. It acts as a cloud-native replacement for local file systems or traditional S3-like services.

  - File uploads/downloads using Appwrite SDK
  - Media referencing from Drupal's media library
  - Role-based storage access control

- **Document Database Sync Layer**

  Connects Drupal's structured content with Appwrite's document database. This allows **storing content nodes** as Appwrite documents for external use or backup, and optionally syncing changes.

  - One-way sync from Drupal → Appwrite
  - JSON serialization of content entities
  - Field mappings defined via admin configuration

- **Configuration Interface**

  Built using Drupal's Form API, this admin-friendly UI **manages all Appwrite credentials, endpoints**, and toggles for enabling/disabling each feature.

  - OAuth provider settings
  - Storage and bucket configuration
  - Document sync enablement options

- **Modular Architecture Design**

All core integrations (Auth, Storage, DB Sync) are implemented as decoupled Drupal services, allowing site builders to enable only the features they need.

  - Follows Drupal's Dependency Injection system
  - Future extensibility: Functions, Realtime, Messaging
  - Clean service-provider model for each Appwrite feature

- **Logging and Debugging Layer (Planned)**

  To support smooth deployment and debugging, the architecture includes plans for an **optional Appwrite log monitor** within Drupal.

  - API request logs
  - Sync success/failure logs
  - Admin-only debug console

- **Development Stack**
  - Frontend/Admin UI: Drupal Form API + JavaScript
  - Backend Logic: PHP (Drupal 10 compatible), REST-based Appwrite SDK
  - Project Tools: GitHub, Trello, Slack, Postman

This modular integration approach brings enterprise-level backend flexibility to Drupal developers, while maintaining the familiarity and control of Drupal's administrative environment.

Community Interaction Plan

Engaging with the broader open-source and Drupal communities is a key focus of this project. Active communication will help ensure that the Appwrite Integration Module aligns with community expectations and remains maintainable and extensible in the long term. Regular interaction with mentors and contributors will help refine the module's design and implementation as it evolves.

During the Community Bonding phase, I will prioritize gathering feedback on architecture decisions, OAuth flow structure, and preferred Appwrite features to support. I'll also participate in existing Drupal channels and forums to stay aligned with the Drupal ecosystem's best practices. Once development begins, consistent documentation and transparency will guide each sprint.

- **Regular Updates:** Weekly progress reports and sync-ups with mentors via Slack or email.

- **Feedback Loops:** Collect input from the community on design and implementation decisions during bonding and early stages.
- **Public Documentation:** Share detailed technical blogs summarizing challenges, implementation insights, and milestone achievements.
- **Long-Term Involvement:** Contribute to code reviews and discussions beyond GSoC to support ongoing development and bug resolution.
- **Educational Resources:** Create a getting-started guide and code walkthroughs to help future contributors build on this work.

Deployment Plan

The Appwrite Integration Module is designed to be lightweight, scalable, and easy to deploy across diverse Drupal environments. It follows a modular structure that allows independent activation of authentication, storage, or document database features depending on site requirements. The deployment plan ensures the module is production-ready and integrates seamlessly with CI/CD pipelines.

The solution will focus on reproducibility and modern tooling, with emphasis on community-friendly development workflows and transparency in deployment.

- **Drupal Module Repository:** The core module will be made available via a public GitHub repository and eventually submitted to Drupal's module directory.
- **Composer Support:** Easy installation via Composer and support for Drupal 10 and above.
- **CI/CD Integration:** GitHub Actions workflows for automated linting, testing, and packaging.
- **Live Demo (Optional):** A sample Drupal site showcasing Appwrite integrations (auth + file storage) hosted on a platform like Render or Pantheon for demonstration purposes.
- **Documentation Site:** Setup using Docusaurus or MkDocs with modular guides, FAQs, and configuration walkthroughs.
- **Local Development:** Provide a Docker-based setup for contributors to run Drupal and Appwrite locally with minimal friction.

This approach ensures rapid iteration, real-time capability, and long-term scalability.

Risks & Mitigation

Every project faces challenges, and this module is no exception. Below is a proactive assessment of potential technical and logistical risks along with the corresponding mitigation strategies to ensure a smooth and successful development journey.

Potential Risk	Mitigation Strategy
OAuth compatibility issues with Drupal's auth system	Leverage Drupal's authentication hooks and user services; test Appwrite login flows early in a sandbox environment.
Evolving Appwrite APIs breaking existing functionality	Pin Appwrite SDK versions; use abstraction layers for API interaction to decouple changes.
Conflicts in media file handling between Drupal and Appwrite	Offer admin-level configuration to toggle between native Drupal media management and Appwrite storage.
Token refresh failures or expired sessions	Implement secure token lifecycle handling using Appwrite's refresh tokens; hook into Drupal cron for background refresh.
Cross-system debugging complexity	Integrate structured logging and a debug mode in the Drupal admin panel to trace API calls and error codes.
Low adoption due to complexity or poor onboarding	Provide rich documentation, example implementations, and beginner-friendly guides to simplify adoption.

Working time - how many hours per week do you plan to work, and how will you divide your time?

I am currently in my second year of college and have carefully planned my academic responsibilities to ensure they do not interfere with my commitment to GSoC. During the community bonding period, I will be available to actively engage in discussions, planning, and initial setup tasks. While my end-semester exams will commence from the first week of May, I will progressively increase my availability as they conclude. From mid-May onward, I will be fully available to dedicate focused time to the project.

Throughout the coding period, I am committed to contributing approximately **40 hours per week**. My schedule is flexible, and I am prepared to allocate additional time as needed to meet project milestones and respond promptly to mentor feedback.

To ensure no disruption in progress, I will **compensate by completing the scheduled work in the prior week**, maintaining steady project development.

Schedule Overview:

- **May (Community Bonding Period)**
    - Available during the first two weeks for planning, discussions, and setup tasks.
    - Full availability from the second half of May after end-semester exams conclude.
- **Coding Period (Mid-May to Early August)**
    - Dedicated 40 hours per week to development, documentation, and regular mentor communication.
- **Mid-Semester Exams (Late August & Late September)**
    - Three days of academic engagement during each exam week.
    - Workload will be adjusted and completed in advance to ensure continuity and avoid delays.

This schedule ensures consistent progress throughout the program and reflects my commitment to delivering high-quality, timely contributions to the project.

Do you have any other plans for the work period (school work, another job, planned vacation)? If so, how do you plan to combine them with your work?

Apart from my academic responsibilities, I have **no other engagements**—no part-time employment or planned vacations—during the GSoC work period. I am fully committed to dedicating my **undivided attention and effort** to the project, ensuring consistent progress and timely delivery throughout the duration.

Future Work / After GSoC Period

The end of the GSoC timeline will mark the beginning of continued development and collaboration. I intend to actively maintain and grow the module to ensure long-term impact and usability within the Drupal and Appwrite communities. My post-GSoC goals include technical enhancements, user-driven improvements, and stronger community engagement.

- **Extend Functionality:** Continue developing advanced features such as Appwrite Function integration, media syncing enhancements, and multi-tenant configurations.
- **Stability & Testing:** Refine error handling, test coverage, and performance across different Drupal environments.
- **Documentation & Tutorials:** Expand educational resources, including example use cases, walkthroughs, and video tutorials to simplify adoption.
- **Community Engagement:** Stay active in forums, respond to issues, and help guide new contributors through onboarding and mentorship.
- **Contribute Back:** Upstream improvements and patterns that emerge from this project will be proposed as reusable tools or libraries for the Drupal ecosystem.

Closing Statement

The Appwrite Integration Module for Drupal is a powerful step toward **modernizing CMS backends** with **scalable, open-source infrastructure**. With my experience in full-stack development, API integrations, and platform architecture, I'm confident in contributing effectively to this project. I'm **excited to collaborate** with mentors and the Drupal community to deliver a meaningful and lasting solution.

Thank you for considering my application!