### **Task 1: Subscription Management**

#### **Subtask 1.1: Integrate with WooCommerce Subscriptions API**

Task 1.1.1: Set up API endpoints for handling subscription lifecycle events

* Research WooCommerce Subscriptions API endpoints
  + Find Documentation: Search for the official WooCommerce Subscriptions API documentation.
  + Read and Summarize: Summarize key points relevant to subscription lifecycle events (creation, renewal, cancellation, upgrade/downgrade).
  + Save Information: Save the summarized research information into a file named subscription\_api\_research.txt located in /documentation/.
  + Save and Commit: Save your research file and update the plugin archive.
* Develop endpoint for subscription creation
  + Use Research: Refer to the saved research file subscription\_api\_research.txt.
  + Define Endpoint: Define the API endpoint URL for subscription creation.
  + Request Format: Document the required request parameters and format.
  + Response Format: Document the expected response structure.
  + Create Endpoint: Implement the API endpoint in the plugin code file at /includes/class-subscription-api.php.
  + Save and Commit: Save your changes and update the plugin archive.
* Develop endpoint for subscription renewal
  + Use Research: Refer to the saved research file subscription\_api\_research.txt.
  + Define Endpoint: Define the API endpoint URL for subscription renewal.
  + Request Format: Document the required request parameters and format.
  + Response Format: Document the expected response structure.
  + Create Endpoint: Implement the API endpoint in the plugin code file at /includes/class-subscription-api.php.
  + Save and Commit: Save your changes and update the plugin archive.
* Develop endpoint for subscription cancellation
  + Use Research: Refer to the saved research file subscription\_api\_research.txt.
  + Define Endpoint: Define the API endpoint URL for subscription cancellation.
  + Request Format: Document the required request parameters and format.
  + Response Format: Document the expected response structure.
  + Create Endpoint: Implement the API endpoint in the plugin code file at /includes/class-subscription-api.php.
  + Save and Commit: Save your changes and update the plugin archive.
* Develop endpoint for subscription upgrade/downgrade
  + Use Research: Refer to the saved research file subscription\_api\_research.txt.
  + Define Endpoint: Define the API endpoint URL for subscription upgrade/downgrade.
  + Request Format: Document the required request parameters and format.
  + Response Format: Document the expected response structure.
  + Create Endpoint: Implement the API endpoint in the plugin code file at /includes/class-subscription-api.php.
  + Save and Commit: Save your changes and update the plugin archive.
* Develop automated tests for subscription creation endpoint
  + Test Plan: Create a test plan for the subscription creation endpoint.
  + Write Tests: Implement the automated tests in the file /tests/test-subscription-creation.php.
  + Run Tests: Execute the tests and document the results.
  + Save and Commit: Save your test files and update the plugin archive.
* Develop automated tests for subscription renewal endpoint
  + Test Plan: Create a test plan for the subscription renewal endpoint.
  + Write Tests: Implement the automated tests in the file /tests/test-subscription-renewal.php.
  + Run Tests: Execute the tests and document the results.
  + Save and Commit: Save your test files and update the plugin archive.
* Develop automated tests for subscription cancellation endpoint
  + Test Plan: Create a test plan for the subscription cancellation endpoint.
  + Write Tests: Implement the automated tests in the file /tests/test-subscription-cancellation.php.
  + Run Tests: Execute the tests and document the results.
  + Save and Commit: Save your test files and update the plugin archive.
* Develop automated tests for subscription upgrade/downgrade endpoint
  + Test Plan: Create a test plan for the subscription upgrade/downgrade endpoint.
  + Write Tests: Implement the automated tests in the file /tests/test-subscription-upgrade-downgrade.php.
  + Run Tests: Execute the tests and document the results.
  + Save and Commit: Save your test files and update the plugin archive.

Task 1.1.2: Implement logic for creating new subscriptions

* Review necessary parameters for creating a subscription
  + Use Research: Refer to the saved research file subscription\_api\_research.txt.
  + Document Parameters: Compile a list of required parameters for creating a subscription.
  + Save Information: Save this parameter list to a file named subscription\_parameters.txt located in /documentation/.
  + Save and Commit: Save this parameter list and update the plugin archive.
* Write the code to handle subscription creation
  + Use Parameters: Refer to the subscription\_parameters.txt file.
  + Develop Logic: Implement the logic to create a subscription using WooCommerce Subscriptions API.
  + Save and Commit: Save the subscription creation code in the file /includes/class-subscription-manager.php.
* Test subscription creation functionality
  + Create Test Cases: Design test cases for various subscription creation scenarios.
  + Execute Tests: Run your tests and document the results.
  + Save and Commit: Save the test results and update the plugin archive.

Task 1.1.3: Develop functionality for renewing subscriptions automatically

* Review renewal mechanics within WooCommerce Subscriptions
  + Use Research: Refer to the saved research file subscription\_api\_research.txt.
  + Document Mechanics: Write down the mechanics of subscription renewal.
  + Save Information: Save this information in a file named renewal\_mechanics.txt located in /documentation/.
  + Save and Commit: Save the renewal mechanics document and update the plugin archive.
* Implement logic to automate subscription renewals
  + Use Mechanics Document: Refer to renewal\_mechanics.txt.
  + Develop Renewal Logic: Write the code to automate subscription renewals.
  + Save and Commit: Save the renewal logic code in /includes/class-subscription-renewal.php.
* Set up cron jobs or WooCommerce scheduling functions
  + Schedule Renewal Jobs: Set up WooCommerce scheduling functions to handle periodic renewals.
  + Save and Commit: Save the scheduling setup in the primary plugin file /custom-subscription-plugin.php.
* Validate automatic renewal functionality in a test environment
  + Create Test Cases: Design test cases for automatic subscription renewals.
  + Execute Tests: Run your tests in a staging environment and document the results.
  + Save and Commit: Save the test results and update the plugin archive.

Task 1.1.4: Implement cancellation logic for subscriptions

* Determine required parameters for subscription cancellation
  + Use Research: Refer to the saved research file subscription\_api\_research.txt.
  + Document Parameters: Compile a list of required parameters for cancellation.
  + Save Information: Save this parameter list in a file named cancellation\_parameters.txt located in /documentation/.
  + Save and Commit: Save this parameter list and update the plugin archive.
* Implement the cancellation logic
  + Use Parameters: Refer to cancellation\_parameters.txt.
  + Develop Logic: Write the logic to handle subscription cancellations, including updating subscription status and notifying the user.
  + Save and Commit: Save the cancellation logic in /includes/class-subscription-cancellation.php.
* Ensure feedback for cancellation actions
  + Develop Feedback Mechanism: Implement user feedback mechanisms (e.g., confirmation messages) in the above-mentioned cancellation class.
  + Save and Commit: Save the changes and update the plugin archive.
* Test cancellation logic across various states
  + Create Test Cases: Design test cases for various subscription states.
  + Execute Tests: Run your tests and document the results.
  + Save and Commit: Save the test results and update the plugin archive.

Task 1.1.5: Develop upgrade/downgrade functionality for subscription plans

* Identify parameters for subscription plan changes
  + Use Research: Refer to the saved research file subscription\_api\_research.txt.
  + Document Parameters: Compile a list of parameters needed for a plan change.
  + Save Information: Save this parameter list in a file named plan\_change\_parameters.txt located in /documentation/.
  + Save and Commit: Save this parameter list and update the plugin archive.
* Implement the logic for upgrading/downgrading subscriptions
  + Use Parameters: Refer to plan\_change\_parameters.txt.
  + Develop Logic: Implement the logic for subscription upgrades/downgrades.
  + Save and Commit: Save the upgrade/downgrade logic in /includes/class-subscription-plan-change.php.
* Adjust billing according to the new plan
  + Billing Adjustment: Ensure that billing is correctly adjusted according to the new subscription plan.
  + Save and Commit: Save the billing adjustment code in /includes/class-subscription-plan-change.php.
* Provide UI for users to change their plans
  + Design UI: Create a user interface that allows users to upgrade/downgrade their subscription plans.
  + Implement UI: Develop the UI components and integrate them into the user dashboard.
  + Save and Commit: Save the UI code in /templates/subscription-plan-change.php.
* Test the upgrade/downgrade functionality
  + Create Test Cases: Design test cases for various upgrade and downgrade scenarios.
  + Execute Tests: Run your tests and document the results.
  + Save and Commit: Save the test results and update the plugin archive.

#### **Subtask 1.2: Custom Subscription Fields (ACF Integration)**

Task 1.2.1: Research ACF integration with WooCommerce Subscriptions to understand capabilities

* Find Documentation: Research ACF and its integration with WooCommerce Subscriptions.
* Documentation Summarize: Summarize the capabilities and limitations.
* Save Information: Save the summarized research in a file named acf\_integration\_research.txt located in /documentation/.
* Save and Commit: Save your research file and update the plugin archive.

Task 1.2.2: Design custom fields to collect additional information during subscription purchase

* Use Research: Refer to the saved research file acf\_integration\_research.txt.
* Design Fields: Create a detailed design of custom fields required.
* Create Mockups: Develop mockups to visualize how these fields will appear on the subscription product page.
* Save Information: Save your mockups and designs in a file named custom\_fields\_design.txt located in /documentation/.
* Save and Commit: Save your design file and update the plugin archive.

Task 1.2.3: Implement ACF fields into subscription product pages and checkout process

* Use Design File: Refer to custom\_fields\_design.txt.
* Add Fields: Implement the custom fields using ACF and integrate them into the subscription product pages and checkout process.
* Data Storage: Ensure the data from these fields is properly stored and associated with the subscription.
* Save and Commit: Save the implementation in /includes/class-subscription-custom-fields.php.

Task 1.2.4: Test subscription workflows with custom fields for accuracy and usability

* Create Test Cases: Design test cases to validate the custom fields in the subscription workflow.
* Execute Tests: Run your tests and gather feedback.
* Usability Testing: Conduct usability tests to ensure the fields enhance the user experience.
* Make Adjustments: Adjust the fields based on feedback.
* Save and Commit: Save any changes and update the plugin archive.

### **Task 2: Membership Levels**

#### **Subtask 2.1: Membership Level Definition**

Task 2.1.1: Design UI for defining membership levels and their benefits

* Design UI: Create wireframes and visual designs for the UI to define membership levels and their benefits.
* Save Information: Save the designs in a file named membership\_levels\_ui\_design.txt located in /documentation/.
* Save and Commit: Save your design file and update the plugin archive.

Task 2.1.2: Implement backend logic to manage membership level configurations

* Develop Logic: Write code to manage membership level configurations in the backend.
* Save and Commit: Save the backend logic in /includes/class-membership-levels.php.

Task 2.1.3: Develop UI components for administrators to manage membership levels

* Implement UI: Develop the UI components for administrators to manage membership levels.
* Save and Commit: Save the UI code in /templates/membership-levels-admin.php.

### **Task 3: Payment Processing**

#### **Subtask 3.1: Payment Gateway Integration**

Task 3.1.1: Integrate with WooCommerce payment gateways (Stripe, PayPal) for secure transactions

* Research Integrations: Gather information on integrating Stripe and PayPal with WooCommerce.
* Save Information: Save the research in a file named payment\_gateway\_research.txt located in /documentation/.
* Save and Commit: Save your research file and update the plugin archive.

Task 3.1.2: Implement secure payment handling for subscription transactions

* Use Research: Refer to the saved research file payment\_gateway\_research.txt.
* Implement Payment Handling: Develop the code to handle payments securely.
* Save and Commit: Save the payment handling code in /includes/class-payment-processing.php.

Task 3.1.3: Test payment workflows in sandbox environments

* Setup Sandbox: Configure sandbox environments for Stripe and PayPal.
* Create Test Cases: Design test cases for payment processing.
* Execute Tests: Run tests and document the results.
* Save and Commit: Save the test results and update the plugin archive.

### **Task 4: Notifications**

#### **Subtask 4.1: Email Notifications**

Task 4.1.1: Configure email notifications for subscription-related events using WooCommerce's built-in system

* Configure Emails: Use WooCommerce's built-in email system to configure notifications for subscription-related events.
* Save and Commit: Save email configuration settings in /includes/class-email-notifications.php.

Task 4.1.2: Define notification triggers and message templates

* Define Triggers: Document the triggers for email notifications.
* Create Templates: Develop email message templates for each trigger.
* Save Information: Save triggers and templates in /documentation/email\_templates.txt.
* Save and Commit: Save your templates file and update the plugin archive.

#### **Subtask 4.2: SMS Notifications (Optional)**

Task 4.2.1: Research and integrate a third-party library for SMS notifications (e.g., Twilio)

* Research SMS Libraries: Gather information on third-party libraries for SMS notifications.
* Save Information: Save the research in a file named sms\_notifications\_research.txt located in /documentation/.
* Save and Commit: Save your research file and update the plugin archive.

Task 4.2.2: Develop logic to trigger SMS notifications for critical subscription events

* Use Research: Refer to the saved research file sms\_notifications\_research.txt.
* Implement SMS Logic: Develop the code to trigger SMS notifications.
* Save and Commit: Save the SMS notification logic in /includes/class-sms-notifications.php.

### **Task 5: Admin Interface**

#### **Subtask 5.1: Admin Dashboard Extension**

Task 5.1.1: Extend WooCommerce admin dashboard to include ASM-specific management tools

* Design Extension: Create a design plan for extending the admin dashboard.
* Save Information: Save the design plan in a file named admin\_dashboard\_extension\_design.txt located in /documentation/.
* Save and Commit: Save your design file and update the plugin archive.

Task 5.1.2: Develop UI components for managing subscribers, membership levels, and subscription settings

* Implement UI: Develop the UI components for managing subscribers, membership levels, and subscription settings.
* Save and Commit: Save the UI components in /templates/admin-dashboard-extension.php.

### **Task 6: User Dashboard**

#### **Subtask 6.1: My Account Section Enhancement**

Task 6.1.1: Design and implement a user-friendly dashboard within the WooCommerce My Account section

* Design Dashboard: Create a design plan for the user dashboard in the My Account section.
* Save Information: Save the design plan in a file named user\_dashboard\_design.txt located in /documentation/.
* Save and Commit: Save your design file and update the plugin archive.
* Implement Dashboard: Develop the user dashboard components as per the design plan.
* Save and Commit: Save the user dashboard components in /templates/account-dashboard.php.

Task 6.1.2: Provide options for subscribers to manage subscriptions, view payment history, and access exclusive content

* Implement Options: Develop options for managing subscriptions, viewing payment history, and accessing exclusive content.
* Save and Commit: Save the options code in /templates/account-dashboard-options.php.

### **Task 7: Content Restriction**

#### **Subtask 7.1: Integration with Content Restriction Library**

Task 7.1.1: Integrate with content restriction libraries like Restrict User Access to enforce access control based on membership levels

* Research Content Restrictions: Gather information on integrating with content restriction libraries.
* Save Information: Save the research in a file named content\_restriction\_research.txt located in /documentation/.
* Save and Commit: Save your research file and update the plugin archive.
* Implement Restriction: Use the research to integrate access control based on membership levels.
* Save and Commit: Save the access control code in /includes/class-content-restriction.php.

Task 7.1.2: Define rules for restricting access to specific content types based on membership levels

* Define Rules: Document rules for restricting access to specific content types.
* Save Information: Save the rules in a file named content\_restriction\_rules.txt located in /documentation/.
* Save and Commit: Save your rules document and update the plugin archive.

### **Task 8: Digital Product Integration**

#### **Subtask 8.1: Support for Multiple File Variations (Optional)**

Task 8.1.1: Explore ACF integration for managing multiple file variations based on subscription levels

* Research ACF Integration: Explore ACF integration for managing multiple file variations.
* Save Information: Save the research in a file named acf\_file\_variations\_research.txt located in /documentation/.
* Save and Commit: Save your research file and update the plugin archive.

Task 8.1.2: Design custom fields to associate digital files with subscription tiers

* Use Research: Refer to the saved research file acf\_file\_variations\_research.txt.
* Design Fields: Create a detailed design of custom fields to associate digital files with subscription tiers.
* Save Information: Save the designs in a file named digital\_files\_custom\_fields\_design.txt located in /documentation/.
* Save and Commit: Save your design file and update the plugin archive.

Task 8.1.3: Implement logic to deliver appropriate files based on subscriber's membership level

* Implement Logic: Develop the logic to deliver appropriate files based on the subscriber's membership level.
* Save and Commit: Save the delivery logic in /includes/class-digital-files-management.php.

### **Task 9: Elementor Integration**

#### **Subtask 9.1: Develop Custom Elementor Widgets**

Task 9.1.1: Research Elementor integration with WooCommerce and ASM plugin

* Find Documentation: Research Elementor integration with WooCommerce and ASM.
* Read and Summarize: Summarize the key points relevant to integrating custom widgets.
* Save Information: Save the summarized research in a file named elementor\_integration\_research.txt located in /documentation/.
* Save and Commit: Save your research file and update the plugin archive.

Task 9.1.2: Design and develop custom Elementor widgets for subscription forms, user dashboards, and subscription statistics

* Use Research: Refer to the saved research file elementor\_integration\_research.txt.
* Design Widgets: Create a detailed design of custom widgets required.
* Develop Widgets: Implement the custom widgets for subscription forms, user dashboards, and subscription statistics.
* Save and Commit: Save the widgets code in /includes/class-elementor-widgets.php.

Task 9.1.3: Test widget functionality and compatibility with Elementor

* Create Test Cases: Design test cases to validate widget functionality and compatibility with Elementor.
* Execute Tests: Run your tests and gather feedback.
* Save and Commit: Save any changes and update the plugin archive.

### **Task 10: Automated Email Campaigns**

#### **Subtask 10.1: Campaign Automation**

Task 10.1.1: Set up automated email campaigns triggered by specific subscriber actions or events

* Configure Automation: Set up automated email campaigns triggered by specific subscriber actions or events.
* Save and Commit: Save the automation settings in /includes/class-email-campaigns.php.

Task 10.1.2: Define workflows for welcome emails, renewal reminders, personalized recommendations, and win-back campaigns

* Define Workflows: Create detailed workflows for welcome emails, renewal reminders, personalized recommendations, and win-back campaigns.
* Save Information: Save the workflows in a file named email\_campaign\_workflows.txt located in /documentation/.
* Save and Commit: Save your workflows document and update the plugin archive.

#### **Subtask 10.2: Personalization**

Task 10.2.1: Customize email content and messaging based on subscriber data and behavior

* Implement Personalization: Customize email content and messaging based on subscriber data and behavior.
* Save and Commit: Save the personalization logic in /includes/class-email-personalization.php.

#### **Subtask 10.3: Segmentation**

Task 10.3.1: Segment subscriber lists based on demographics, interests, and engagement levels

* Define Segments: Document the criteria for segmenting subscriber lists based on demographics, interests, and engagement levels.
* Save Information: Save the segmentation criteria in a file named subscriber\_segmentation.txt located in /documentation/.
* Save and Commit: Save your segmentation document and update the plugin archive.

Task 10.3.2: Target email campaigns to specific subscriber segments for improved relevance and engagement

* Implement Segmentation: Develop the code to target email campaigns to specific subscriber segments.
* Save and Commit: Save the segmentation logic in /includes/class-email-segmentation.php.

#### **Subtask 10.4: Integration**

Task 10.4.1: Integrate with email marketing platforms (e.g., Mailchimp, Klaviyo) for seamless campaign management

* Research Email Marketing Integrations: Explore integration with email marketing platforms.
* Save Information: Save the research in a file named email\_marketing\_integration\_research.txt located in /documentation/.
* Save and Commit: Save your research file and update the plugin archive.

Task 10.4.2: Ensure compatibility and integration with popular email marketing tools

* Implement Integration: Develop the logic to integrate with popular email marketing tools.
* Save and Commit: Save the integration code in /includes/class-email-integration.php.

### **Task 11: Documentation and Support**

#### **Subtask 11.1: Documentation**

Task 11.1.1: Create comprehensive documentation for developers, administrators, and end-users

* Write Documentation: Create comprehensive documentation for developers, administrators, and end-users.
* Save and Commit: Save the documentation in /documentation/.

Task 11.1.2: Document installation instructions, configuration options, and troubleshooting steps

* Write Instructions: Create installation instructions, configuration options, and troubleshooting steps.
* Save and Commit: Save the instructions in /documentation/.

#### **Subtask 11.2: Testing**

Task 11.2.1: Develop a testing plan covering unit tests, integration tests, and user acceptance tests

* Write Test Plan: Develop a testing plan covering unit tests, integration tests, and user acceptance tests.
* Save and Commit: Save the test plan in /documentation/test\_plan.txt.

Task 11.2.2: Conduct thorough testing to identify and fix any bugs or issues

* Conduct Tests: Run comprehensive tests to identify and fix any bugs or issues.
* Save and Commit: Save the test results in /documentation/test\_results/.

### **Task 12: Deployment and Support**

#### **Subtask 12.1: Deployment**

Task 12.1.1: Prepare the plugin for deployment to WordPress.org or other distribution platforms

* Prepare Package: Compile and prepare the plugin for deployment.
* Save and Commit: Ensure the deployment package is saved and structured correctly in /.

Task 12.1.2: Test the deployment process and ensure compatibility with WordPress standards

* Deploy Test: Test the deployment process to ensure compatibility with WordPress standards.
* Save and Commit: Save the test results and ensure the deployment process is documented in /documentation/deployment\_instructions.txt.

#### **Subtask 12.2: Support**

Task 12.2.1: Set up channels for user support such as forums, email, or ticketing systems

* Create Channels: Set up user support channels such as forums, email, or ticketing systems.
* Save Information: Document the support channels in /documentation/user\_support.txt.
* Save and Commit: Save the support documentation and update the plugin archive.

Task 12.2.2: Provide ongoing support and maintenance to address user queries and issues promptly

* Establish Support Protocol: Develop a protocol for ongoing support and maintenance.
* Save and Commit: Save the support protocol in /documentation/maintenance\_protocol.txt.