Intimate Tales - Code Generation Plan

YAML Description of New Files

```
- intimate-tales-plugin/intimate-tales.php:
    purpose: 'Plugin configuration and initialization.'
    variablesExported:
        - IT_VERSION
        - IT_PLUGIN_PATH
        - IT_PLUGIN_URL
        - IT_PLUGIN_BASENAME
        - IT_ACF_PATH
        - IT_ENQUEUE_PREFIX
        - IT_TEXTDOMAIN
        - IT_STORY_POST_TYPE
        - IT_HOOK_PREFIX
        - IT_CACHED_STORY_RESULTS
        - IT_CACHED_USER_DATA
        - IT_AUTHENTICATE_NONCE
        - IT_PAIRING_REQUEST_NONCE
        - IT_STORY_ACTION_NONCE

- intimate-tales-plugin/acf-json/option-pages/user-settings.json:
      purpose: 'Store settings related to users.'
      dataSchema:
          - display_name
          - profile_image
          - notification_settings

- intimate-tales-plugin/acf-json/option-pages/design-settings.json:
      purpose: 'Settings related to design and UI.'
      dataSchema:
          - theme_color
          - font_style

- intimate-tales-plugin/acf-json/option-pages/interaction-
settings.json:
      purpose: 'Settings related to user interaction.'
      dataSchema:
          - interactions_limit
          - reaction_types

- intimate-tales-plugin/acf-json/option-pages/payment-settings.json:
      purpose: 'Store settings related to payment options and
configurations.'
      dataSchema:
          - currency_type
          - payment_gateway
          - transaction_fee

- intimate-tales-plugin/acf-json/option-pages/smtp-settings.json:
    purpose: 'Settings for SMTP configurations.'
    dataSchema:
        - smtp_host
        - smtp_user
        - smtp_pass

- intimate-tales-plugin/includes/classes/class-main.php:
    purpose: 'Handles core functionalities of the IntimateTales
Plugin.'
    methods:
        - on_activation()
        - on_deactivation()
        - load_textdomain()
        - enqueue_scripts_and_styles()
        - run()

- intimate-tales-plugin/includes/classes/class-template-loader.php:
    purpose: 'Handles the dynamic loading of templates, allowing for
theme overrides and fallback to the plugin’s default templates.'
    properties:
        - template_path: 'Path for the template file. Defaults to the
plugin’s template directory.'
    methods:
        - get_template_part($slug, $name = null, $args = [], $subdir
= ''): 'Retrieves the template part. Similar to WordPress
`get_template_part` function but allows for passing arguments to the
template and specify a sub-directory where the template is located.'
        - load_template($template_name, $args = [], $subdir = ''):
'Loads a specified template, checks if a theme override exists,
otherwise defaults to the plugin’s template. Passes arguments to the
template and allows specification of a sub-directory where the
template might be located.'
        - locate_template($template_name, $subdir = ''): 'Locates the
template, checks if a theme override exists, otherwise defaults to
the plugin’s template directory. Also checks within an optional sub-
directory.'

- intimate-tales-plugin/includes/classes/class-authentication.php:
    purpose: 'Handles user authentication. The class now supports
template overrides and loads settings from config and ACF JSON if
available.'
    properties:
        - is_two_factor_enabled: 'Indicates whether two-factor
authentication is enabled.'
    methods:
        - login_user()
        - register_user()
        - logout_user()
        - authenticate_user($username, $password): 'Verifies username
and password. Returns a user object if valid, null otherwise.'

- intimate-tales-plugin/includes/classes/class-shortcodes.php:
    purpose: 'Handles the creation and rendering of plugin
shortcodes.'
    shortcodesRegistered:
        - it_login_form
        - it_signup_form
        - it_pw_reset_form
    methods:
        - render_login_form(): 'Renders the login form using the
shortcode.'
        - render_signup_form(): 'Renders the signup form using the
shortcode.'
        - render_pw_reset_form(): 'Renders the password reset form
using the shortcode.'

- intimate-tales-plugin/includes/classes/class-security.php:
    purpose: 'Central class to handle all security-related
operations.'
    methods:
        - hash_password($password): 'Returns a hashed version of the
provided password.'
        - verify_password($password, $hash): 'Verifies a password
against its hash.'
        - generate_nonce($action): 'Generates a nonce for a specific
action.'
        - verify_nonce($nonce, $action): 'Verifies the nonce for the
specified action.'

- intimate-tales-plugin/includes/classes/class-feedback.php:
    purpose: 'Manages feedback collection and processing.'
    methods:
        - collect_user_feedback(): 'Presents a feedback form to the
user.'
        - process_feedback_submission($data): 'Processes and stores
feedback from users.'

- intimate-tales-plugin/includes/classes/class-dashboard.php:
    purpose: 'Handles dashboard interactions.'
    idNames:
        - dashboard_container
        - recent_stories_section
        - user_profile_section
    messageNames:
        - DASHBOARD_LOAD_SUCCESS
        - DASHBOARD_LOAD_ERROR
    functions:
        - load_dashboard()
        - update_user_info()

- intimate-tales-plugin/includes/classes/class-roleplay-
interface.php:
    purpose: 'Manage the roleplay interactions.'
    idNames:
        - roleplay_container
        - decision_point_button
    methods:
        - initiate_roleplay()
        - save_roleplay_decision()
- intimate-tales-plugin/includes/classes/class-woocommerce-
integration.php:
    purpose: 'Integrate WooCommerce functionalities into the
IntimateTales platform.'
    properties:
        - woocommerce_options: 'Holds settings related to WooCommerce
integration.'
    methods:
        - initialize_integration(): 'Sets up and initializes the
WooCommerce functionalities.'
        - manage_product_addition($product_data): 'Handles the
addition of new products.'
        - manage_order_process($order_data): 'Handles the order
processes and potential custom workflows related to IntimateTales.'

- intimate-tales-plugin/includes/utilities-functions.php:
    purpose: 'Utility functions used across the plugin for various
generic tasks.'
    functions:
        - it_generate_random_string($length): 'Generates a random
string of a specified length.'
        - it_format_date($date, $format): 'Formats a date according
to the specified format.'
        - it_sanitize_input($input): 'Sanitizes input data to prevent
security vulnerabilities.'

- intimate-tales-plugin/includes/classes/class-notification.php:
    purpose: 'Send notifications to users.'
    messageNames:
        - NOTIFICATION_SENT
        - NOTIFICATION_ERROR
    methods:
        - send_email_notification()
        - send_sms_notification()

- intimate-tales-plugin/includes/classes/class-pairing-system.php:
      purpose: 'Manage the user pairing/friend request system.'
      idNames:
          - send_pair_request_button
          - accept_pair_request_button
          - decline_pair_request_button
      functions:
          - send_pair_request()
          - accept_pair_request()
          - decline_pair_request()

- intimate-tales-plugin/includes/classes/class-ai-story-
generator.php:
      purpose: 'Generate stories using AI algorithms.'
      functions:
          - generate_story()
          - save_generated_story()

- intimate-tales-plugin/includes/classes/class-gamification.php:
      purpose: 'Handle daily challenges and competitions.'
```

```
        idNames:
            - challenge_list
            - monthly_competition_leaderboard
        messageNames:
            - CHALLENGE_COMPLETED
            - NEW_CHALLENGE_AVAILABLE
        functions:
            - update_challenge_status()
            - load_leaderboard()

  - intimate-tales-plugin/includes/classes/class-rewards.php:
        purpose: 'Manage user rewards and points.'
        idNames:
            - user_points_display
            - redeem_reward_button
        functions:
            - add_points_to_user()
            - redeem_reward()

  - intimate-tales-plugin/includes/classes/class-onboarding.php:
        purpose: 'Handle the onboarding process for new users.'
        idNames:
            - registration_form
            - setup_profile_form
            - tutorial_section
        functions:
            - register_new_user()
            - setup_user_profile()
            - display_tutorial()

  - intimate-tales-plugin/includes/classes/class-category-list.php:
        purpose: 'Display and manage story categories.'
        idNames:
            - category_list
            - search_categories_input
        functions:
            - display_all_categories()
            - search_for_category()

  - intimate-tales-theme/functions.php:
        purpose: 'Theme setup and initial configurations.'
        functions:
            - intimate_tales_theme_setup()

  - intimate-tales-theme/assets/scss/style.scss:
        purpose: 'Main stylesheet for the theme.'
        variablesExported:
            - primary_color
            - secondary_color

  - intimate-tales-theme/single-story.php:
        purpose: 'Display single story content.'
        idNames:
            - story_content
            - story_comments_section
```

```
- intimate-tales-theme/page-dashboard.php:
      purpose: 'Dashboard layout.'
      idNames:
          - user_stories_list
          - user_notifications

- intimate-tales-theme/page-onboarding.php:
      purpose: 'Onboarding page layout.'
      idNames:
          - onboarding_steps
          - finish_onboarding_button

- intimate-tales-theme/template-partss/footer.php:
      purpose: 'Site footer template.'
      idNames:
          - footer_navigation
          - site_credits

- intimate-tales-theme/template-parts/sidebar.php:
      purpose: 'Sidebar template for additional site features.'
      idNames:
          - sidebar_widgets
          - recent_stories_widget

- intimate-tales-theme/template-parts/header.php:
      purpose: 'Site header template.'
      idNames:
          - main_navigation
          - search_bar

- intimate-tales-theme/template-parts/dark-mode-toggle.php:
      purpose: 'Dark mode toggle feature.'
      idNames:
          - dark_mode_switch
      functions:
          - toggle_dark_mode()

- intimate-tales-plugin/template-parts/auth/login-form.php:
    purpose: 'Template to display the login form.'
    formInputs:
        - username
        - password
    formActions:
        - custom_login
    submissionMethod: POST
    endpoint: admin-post.php

- intimate-tales-plugin/template-parts/auth/signup-form.php:
    purpose: 'Template to display the signup form.'
    formInputs:
        - username
        - email
        - password
    formActions:
        - custom_signup
    submissionMethod: POST
    endpoint: admin-post.php

- intimate-tales-plugin/template-parts/auth/password-reset-form.php:
    purpose: 'Template to allow users to request a password reset.'
    formInputs:
        - email
    formActions:
        - custom_pw_reset
    submissionMethod: POST
    endpoint: admin-post.php
```

Optimization Plan

Optimization of the code will involve several steps:

1. **Refactoring**: Regularly review the code to ensure clarity and simplicity. Especially with multiple developers, maintaining a consistent coding style and structure is essential.
2. **Removing Unused Code**: As the plugin evolves, it's common for certain features or functions to become obsolete. Periodically, audit the code to identify and remove any unused or deprecated segments.
3. **Improving Performance**:
    - Database Optimization: Ensure that database queries are optimized and not creating unnecessary overhead.
    - HTTP Requests: Minimize the number of requests by combining scripts, styles, and making use of asynchronous loading when feasible.
    - Caching: Utilize caching mechanisms like transient APIs in WordPress to reduce redundant operations. Implementing object caching or page caching could also be beneficial.
    - Asset Optimization: Minimize and compress CSS, JS, and images.

Expansion Plan

Expansion of the code will involve adding new features and improving existing ones. This could involve:

1. **Adding New Features**:
    - AI-driven Enhancements: Expand on the AI story generator for more intricate and diverse storylines.
    - User Feedback: Regularly collect and evaluate user feedback to identify areas of improvement or new features that could enhance their experience.
    - Integration with Other Systems: Depending on the growth, consider integrations with other platforms or third-party systems to expand functionality.

2. **Improving Existing Features**:
    - UI/UX Enhancements: Regularly review and update the user interface based on user feedback and modern design trends.
    - Performance Improvements: Continuously monitor the performance and optimize areas causing bottlenecks or slowdowns.
    - Accessibility: Ensure the platform remains accessible to all users, including those with disabilities.
3. **Planning for Scalability**:
    - Code Structure: Ensure that the codebase is structured in a way that allows for easy expansion and modular additions.
    - Database Scalability: Plan for growth in database size and ensure efficient querying mechanisms. This includes periodically optimizing database tables.
    - Hosting and Infrastructure: As traffic grows, consider scalable hosting solutions or even moving to cloud-based infrastructures which can handle spikes in traffic and offer easy scalability.

Documentation and Training

To ensure that the team is aligned with the codebase, and for easier onboarding of new developers:

1. Code Comments: Ensure each function, method, and class has descriptive comments detailing their purpose, parameters, and expected return values.
2. Technical Documentation: Create and maintain a technical documentation that provides an overview of the codebase structure, describes the purpose of different classes and files, and provides examples of how to use the main functions and methods.
3. Training Sessions: Organize regular training sessions to bring new team members up to speed and to provide updates on any new coding practices or technologies being introduced.

Quality Assurance

1. Automated Testing: Implement unit and integration tests using tools like PHPUnit. This ensures that new code additions or changes don't unintentionally break existing functionality.
2. Code Review: Establish a code review process where every code change is reviewed by at least one other developer before merging. This not only helps catch potential bugs but also ensures code quality and consistency.
3. Staging Environment: Set up a staging environment that mirrors the production environment. All new code should be deployed to staging first to ensure it works as expected before being deployed to production.
4. Feedback Loop: Ensure there's an easy way for users to report bugs or issues they encounter. Regularly review this feedback to identify areas for improvement.

Security Measures

1. Data Validation and Sanitization: Always validate and sanitize data coming from user inputs or external sources to prevent SQL injections, cross-site scripting, and other vulnerabilities.
2. Regular Audits: Use tools like WPScan or other vulnerability scanners to regularly audit the plugin for potential security threats.
3. Limit Direct Access: Ensure that none of the plugin files can be directly accessed. This can be done by placing a direct file access check at the top of PHP files.
4. Use Nonces: Utilize WordPress nonces for forms and URLs to ensure the request is coming from a trusted source.
5. Stay Updated: Ensure that all external libraries, tools, or dependencies the plugin relies on are regularly updated to their latest versions, which might contain security patches.