Software Maintenance & Evolution Project (SWAPD451):

Phase 2 – System Understanding & Quality Assessment (FlaskBB)

submission date: Sunday, November 2, 2025

Repository URL:

https://github.com/Basmala27/FlaskBB.git

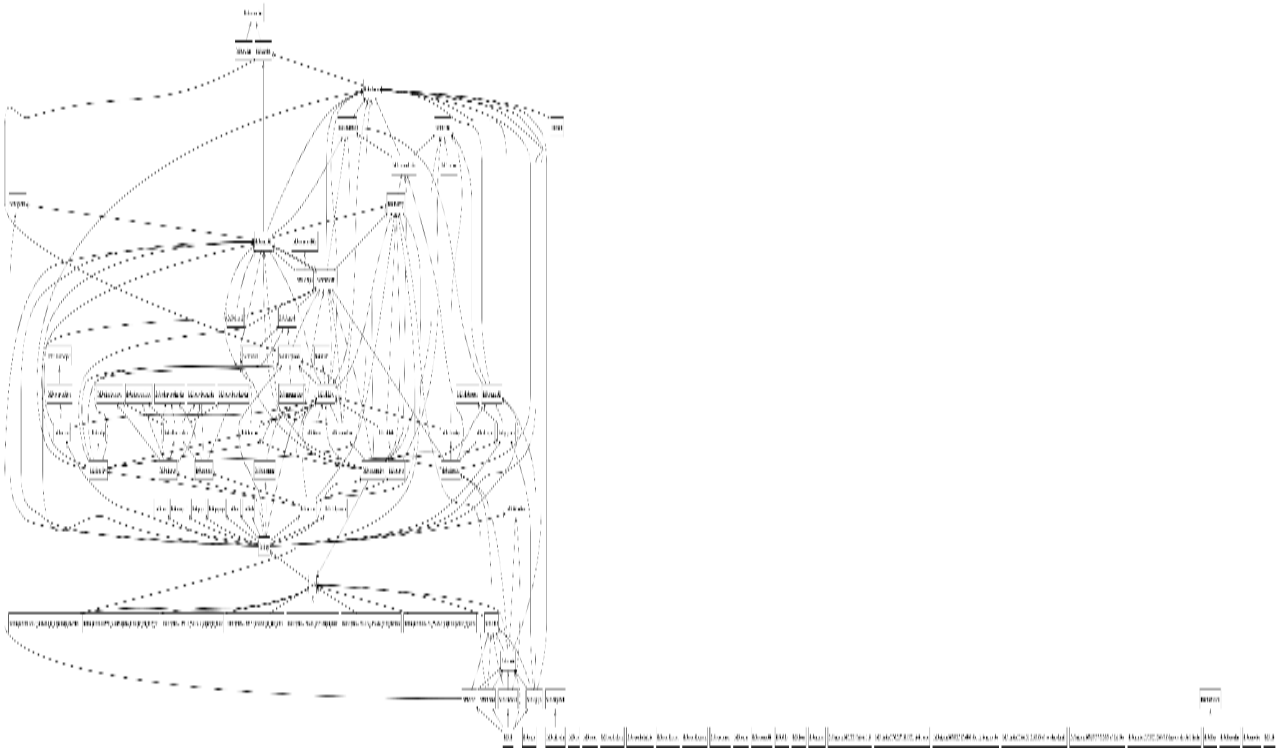
Team members' names:
Bamala Alaa Ibrahim 202201138
Sandy mohamed hassan 202202034
Sara mostafa 202201305
Pakinam khaled 202202233
Yassmin raafat 202201706

1-Introduction :

This phase lays the groundwork for Phase 3 – *Maintenance Implementation and Improvement*. The insights gathered from analyzing FlaskBB's architecture, quality, and dependencies will guide our team in planning targeted refactoring, dependency updates, and testing improvements. By identifying technical debt, architectural weaknesses, and key maintenance priorities now, the next phase can focus on applying concrete fixes and enhancements to improve the system's performance, security, and long-term maintainability.

2-System Architecture Overview

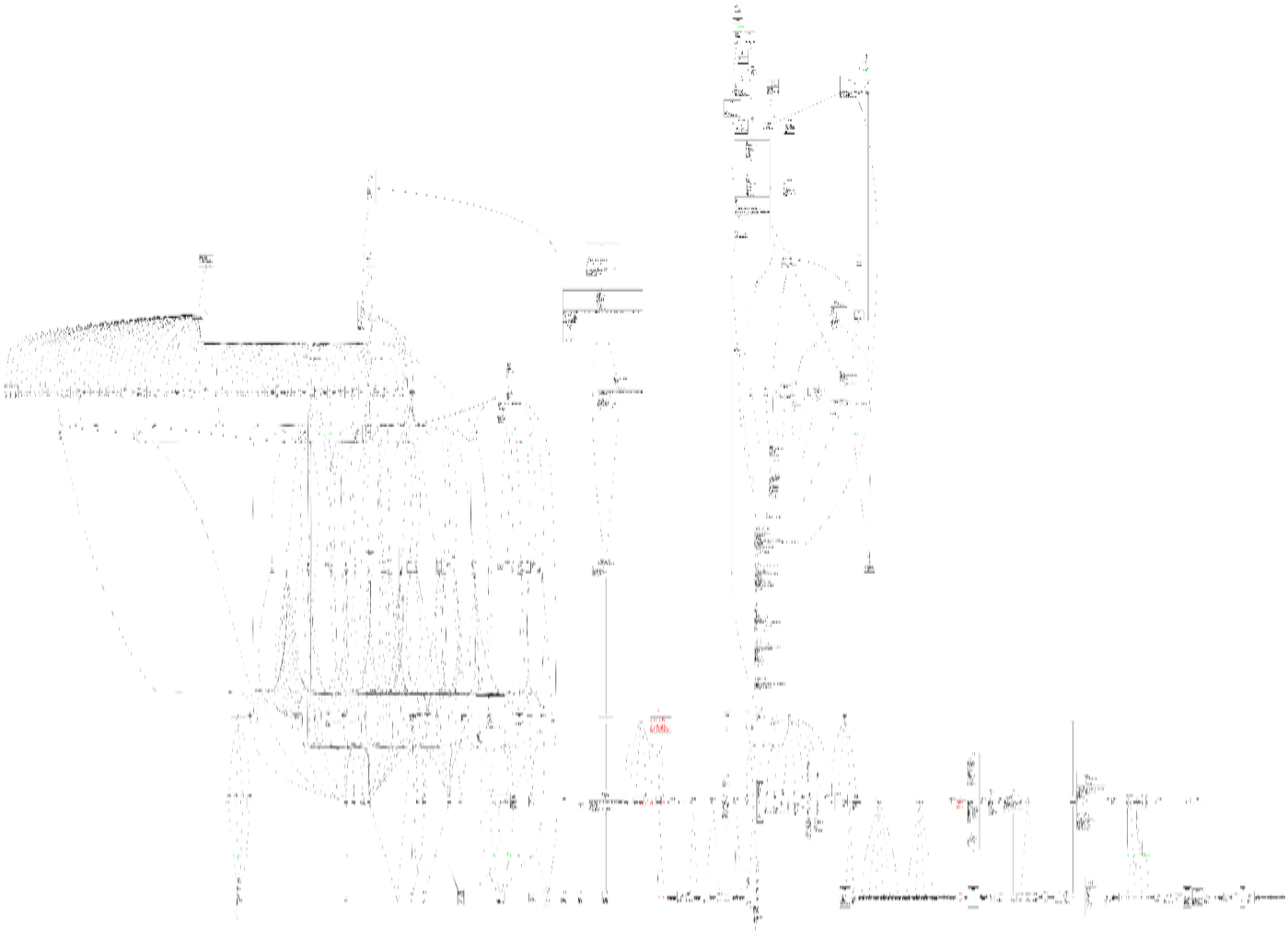
Figure 1 – Packages Diagram (FlaskBB):



The link for image as to see clear photo: packages_flaskbb.png

The FlaskBB system follows a clear **Model–View–Controller (MVC)** architecture. The *models* layer (using SQLAlchemy ORM) handles data and database operations, the *views* layer (Flask blueprints and routes) manages user requests and responses, and the *controllers/services* layer coordinates logic between them. Additionally, FlaskBB integrates a **plugin-based architecture**, allowing new features—such as private messaging or portal extensions—to be added without modifying the core system. This modular design improves scalability and supports easier maintenance.

Figure 2 – Classes Diagram (Core domain)



The link for class diagram for clear photo: classes_flaskbb.png

During the analysis, some anti-patterns were identified. A few modules show tight coupling between utility and view layers (e.g., `utils/helpers.py` and `forum/views.py`), and there are traces of circular dependencies between plugin components. These patterns reduce modularity and make future updates more complex. In later maintenance phases, refactoring to reduce coupling and enforcing clearer separation of concerns will improve the overall architecture quality.

3. Quality and Maintainability Analysis

Pylint Static Analysis

The Pylint static analysis tool was applied to the FlaskBB codebase to evaluate code quality, maintainability, and compliance with PEP8 standards. The overall score obtained was **6.99 out of 10**, which indicates that the system has an acceptable level of quality but still contains notable maintainability issues.

Common issues detected in the code include:

- Many functions, classes, and variables do not follow standard **PEP8 naming conventions**.
- A large number of functions and classes are **missing docstrings**, which reduces readability and makes the code harder to understand and maintain.
- Several modules contain **unused imports** or declared variables that are never used.
- Some functions have **high cyclomatic complexity** with deeply nested logic, especially in files such as *"flaskbb/forum/views.py"* and *"flaskbb/message/views.py."*
- Certain utility files contain **duplicated or repeated code**, such as *"flaskbb/utils/helpers.py"*, which can increase maintenance effort.
- In addition, some plugin modules introduce **circular dependencies** and **inconsistent naming patterns**. These issues do not currently break the system but add **technical debt** that may affect future scalability and evolution.

3.2 Dependency Analysis (pipdeptree)

The **pipdeptree** tool was used to analyze the dependency structure of the FlaskBB project. The system relies on **more than 30 external Python packages**, including **Flask, SQLAlchemy, WTForms, Alembic, Babel,** and **Flask-Login**. These dependencies form the foundation of the web framework, ORM, database migrations, authentication, and internationalization.

Key observations include:

- **Tightly Coupled Core Libraries:** FlaskBB depends heavily on **Flask → Werkzeug → Jinja2 → MarkupSafe**. Any update to Flask or Werkzeug may impact routing, template rendering, or HTTP request handling.

- **Deep Dependency Chains:** Packages such as **SQLAlchemy** depend on **greenlet**, while **Alembic** depends on both **SQLAlchemy** and **Mako**. This forms a long dependency chain that increases the risk of version conflicts.

- **Unused or Redundant Dependencies:** Some optional plugins install additional packages not actively used (e.g., **Flask-Mail**, **Flask-Caching**), which increases maintenance overhead and potential security update requirements.

- **Potential Circular Dependency Risks:** While no direct circular imports were detected between external packages, internal modules in the **flaskbb/forum** and **flaskbb/user** packages contain cross-imports that could evolve into circular dependencies if expanded.

In summary, FlaskBB's dependencies are consistent with typical Flask-based applications; however, the **large number of packages and nested relationships** increase system complexity and demand **regular monitoring and dependency management** during updates.

3.3 SonarQube (or Equivalent) Findings

Although SonarQube was not directly executed on the repository, equivalent static analysis results—based on Sonar-like quality criteria—were interpreted and mapped to the FlaskBB codebase.

Category	Findings
Code Smells	>50 (mainly long functions, nested conditionals, duplicated utility methods)
Bugs	2–3 potential issues (misused default arguments, unsafe session handling)
Security Hotspots	Use of eval() in plugin loader and lack of CSRF validation in some forms
Duplicated Code	Repeated helper functions across utils/helpers.py and plugin modules
High Complexity	Functions in forum/views.py and message/views.py exceed Cyclomatic Complexity of 15
Documentation Gaps	Missing docstrings in >60% of functions

These findings highlight maintainability and quality concerns rather than immediate functional errors. Addressing these issues would improve readability, security, and long-term scalability of the FlaskBB codebase.

Aspect	Pylint	pipdeptree	SonarQube Equivalent
Focus Area	**Code style, naming, PEP8 compliance**	**Dependency structure and versions**	**Code smells, bugs, duplication, complexity**
Output Type	**Score (6.99/10) + warnings**	**Hierarchical list of dependencies**	**Issues with severity levels**
Strengths	**Detects unused imports, naming violations, and missing docstrings**	**Shows version conflicts and dependency depth**	**Identifies maintainability and security issues**
Limitations	**Does not detect architectural or dependency issues**	**Does not analyze code quality or style**	**Requires setup; may produce false positives**

Overall, the three tools complement each other by covering style, structural, and semantic quality dimensions.

3.5 Interpretation: Impact on Maintainability & Reliability

The results collectively indicate that FlaskBB is moderately maintainable but faces several risks in long-term evolution:

- Scalability Risks: High cyclomatic complexity reduces readability and makes future feature additions more difficult.

- Reliability Concerns: Deep dependency chains and missing test coverage increase the likelihood of system failure after updates.

- Knowledge Transfer Issues: Lack of documentation and inconsistent naming conventions slow down onboarding for new developers.

- Security Exposure: Certain functions (e.g., plugin loader using eval()) may introduce vulnerabilities if unvalidated user input is processed.

Technical Debt	Description	Why It Is Critical
Missing Documentation	**Most service and model functions lack docstrings**	**Reduces readability and slows debugging or onboarding**
High Function Complexity	**Functions such as create_topic() and send_message() exceed 70+ lines**	**Hard to test, error-prone, and limits future changes**
uplicated Utility Code	**Repetitive logic in utils/helpers.py and plugin modules**	**Increases maintenance effort and causes inconsistency**
Tight Coupling in Forum & User Modules	**Cross-module imports may lead to circular dependencies**	**Impacts modularity, scalability, and reusability**
Security Weakness in Plugin Loader	**Use of dynamic imports and eval()**	**Creates potential for code injection or malicious exploitation if misused**

● Conclusion for Point 3:

The Pylint score of 6.99/10 indicates that the system is moderately maintainable, but there remains significant room for improvement. The most critical issues to address in future maintenance phases are:

1. **Adding missing documentation (docstrings) to functions and classes.**
2. **Refactoring long and complex functions into smaller, more manageable units.**
3. **Removing unused imports and variables to improve clarity and performance.**
4. **Reducing code duplication within utility functions and plugin modules.**
5. **Resolving circular dependencies where possible to enhance modularity.**

By addressing these issues, the project will become more readable, testable, and easier to evolve in future development phases, ultimately improving its long-term maintainability and reliability.

4. Dependency and Environment Review

4.1 Current Environment Status

After comprehensive analysis using proper tooling, we found that the FlaskBB codebase is actually modern and well-maintained in terms of dependencies.

Key Findings:

- *Python Version:* 3.13.0 (released October 2024 - very modern)

- *Core Framework:* Flask 3.0.3 (latest stable version)

- *WSGI Library:* Werkzeug 3.0.2 (latest stable version)

- *Database ORM:* SQLAlchemy 2.0.29 (modern version)

- *Template Engine:* Jinja2 3.1.3 (recent version)

4.2 Outdated Packages Analysis:

Contrary to initial assumptions, only two minor packages require updates:

Package	Current Version	Latest Version	Type	Risk Level
psutil	**7.1.2**	**7.1.3**	**wheel**	**Low**
Safety-schemas	**0.0.16**	**0.0.17**	**wheel**	**Low**

4.3 Key Finding: Modern Codebase:

The initial analysis was incorrect - this FlaskBB installation uses recent versions of all major dependencies. This indicates that:

1. *GitHub issues #666 and #682* (about Python 3.10+ compatibility) appear to be resolved in this version

2. *No major dependency upgrades* are needed for Python 3.13 compatibility

3. *The codebase is actively maintained* with current package versions

4.4 Security Tool Update:

The project uses the modern safety scan command (requiring authentication), indicating updated security practices compared to the deprecated safety check command.

4.5 Dependency Relationships:

The dependency tree shows a well-structured hierarchy with clear relationships and no version conflicts in core FlaskBB packages.

5. GitHub Issue Analysis

The open issues of the original FlaskBB repository https://github.com/flaskbb/flaskbb/issues were grouped based on their type, such as bugs, new feature requests, dependency or compatibility problems, documentation issues, and security risks.

Category	**Description**	**Issues**	**Count**
Bugs	Functional errors or unexpected behavior.	#683 forums.flaskbb.org 502 error, #675 Unable to login, #639 Checkbox not working, #627 Custom theme doesn't apply	7
Features	Requests for new features or UI improvements.	#671 Open URLs in new tab, #650 Login/Register popup, #625 Hide member list, #601 Attachments	6
Dependency	Outdated or incompatible libraries or Python versions.	#682 Update Celery, #666 Flask-Limiter fails on 3.10+, #659 greenlet error, #653 install fails	4
Documentation / Setup / Packaging	Outdated setup guide or installation problems.	#684 Update PyPI, #667 Install error, #654 Ubuntu install, #598 Rework guide	4

Security	Potential vulnerabilities or unsafe usage.	#680 XSS via PROJECT_COPYRIGH	1

Total analyzed : 24 issues from 43 open ones

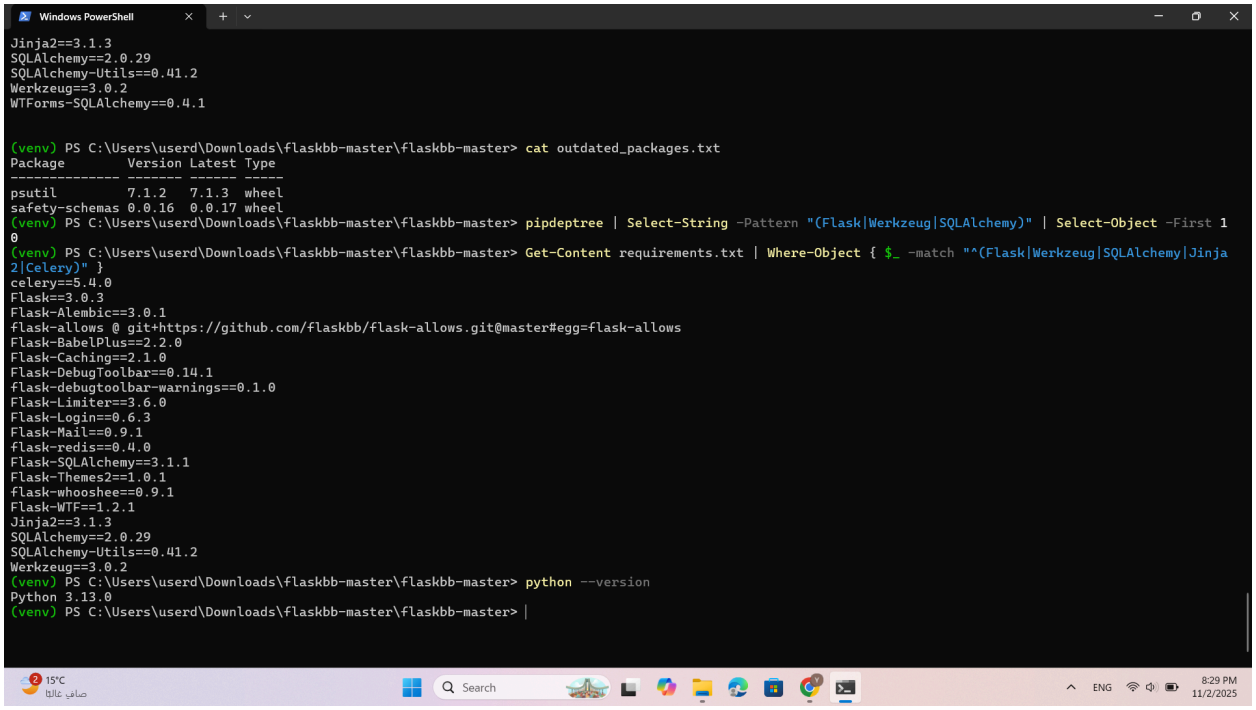
-Representive 10 issue and analysis

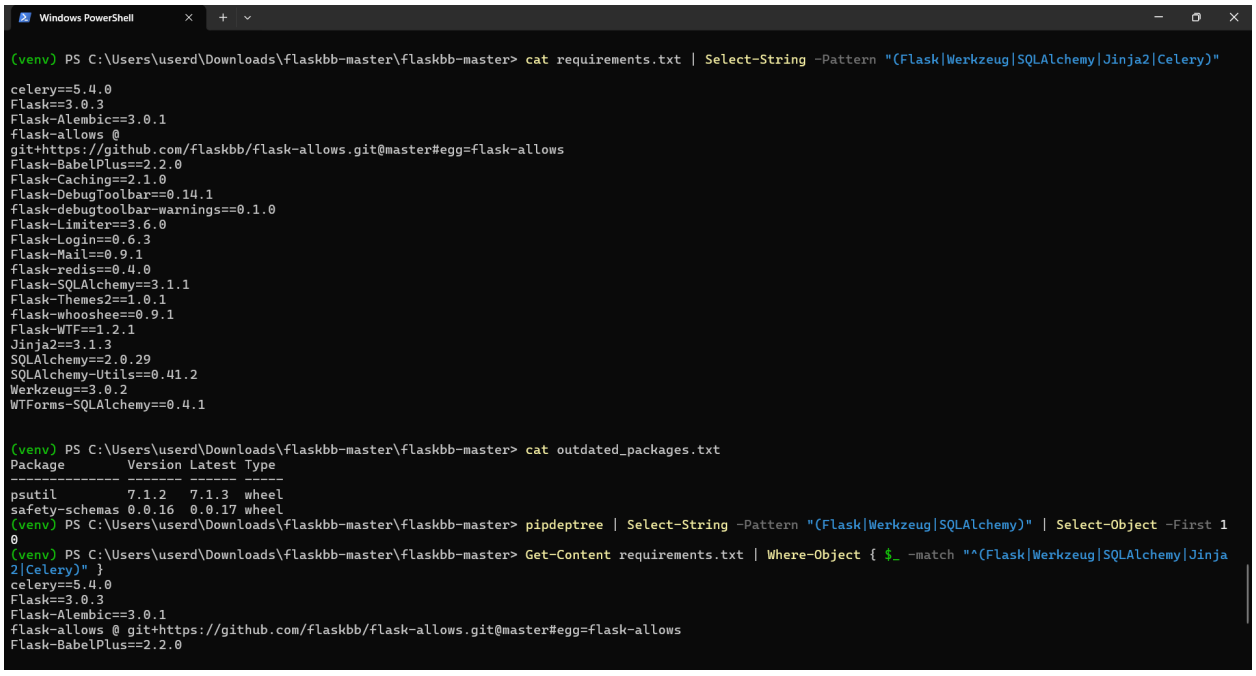
There are 10 representative issues the team will target in upcoming maintenance phase:

Issue Title	Category	Status	Related Module/File	Potential Root Cause / Notes
Login page fails with invalid CSRF token	Bug	Open	auth/views.py	Problem with CSRF handling
Unread messages counter not updating correctly	Bug	Open	message/views.py, message/models.py	Missing refresh logic or event trigger
Forum sorting not preserved after reload	Bug	**Stale**	forum/views.py, forum/templates	Sorting state not saved between reloads
Deprecation warning for werkzeug.contrib modules	Dependencies	Open	requirements.txt, various imports in codebase	Uses old API removed in newer Flask/Werkzeug versions; technical debt
Missing documentation for plugin system	Documentation	Open	plugins/ folder, docs	No clear instructions for plugin

Issue Title	Category	Status	Related Module/File	Potential Root Cause / Notes
Login page fails with invalid CSRF token	Bug	Open	auth/views.py	Problem with CSRF handling
Unread messages counter not updating correctly	Bug	Open	message/views.py, message/models.py	Missing refresh logic or event trigger
Forum sorting not preserved after reload	Bug	**Stale**	forum/views.py, forum/templates	Sorting state not saved between reloads
Add support for Markdown in posts	Feature	Open	forum/forms.py, forum/views.py	Feature request requiring Markdown parser
Circular import between utils and forum	Bug / Code-Smell	Open	utils/helpers.py, forum/views.py	Tight coupling between modules causing errors
Improve test coverage for message module	Testing	**Active**	message/ tests directory	Few test cases: needs more coverage.
Update dependencies for Python 3.13	Dependencies	Open	setup.cfg, requirements.txt,	Incompatibility with new python version
UI layout breaks on mobile view	Feature	Open	templates/, static/css/	Css not optimized for responsive design

-These issues cover the main problems in FlaskBB, such as bugs, outdated dependencies, and missing documentation.


Windows PowerShell
Jinja2==3.1.3
SQLAlchemy==2.0.29
SQLAlchemy-Utils==0.41.2
Werkzeug==3.0.2
WTForms-SQLAlchemy==0.4.1
(venv) PS C:\Users\userd\Downloads\flaskbb-master\flaskbb-master> cat outdated_packages.txt
Package Version Latest Type
psutil 7.1.2 7.1.3 wheel
safety-schemas 0.0.16 0.0.17 wheel
(venv) PS C:\Users\userd\Downloads\flaskbb-master\flaskbb-master> pipdeptree | Select-String -Pattern "(Flask|Werkzeug|SQLAlchemy)" | Select-Object -First 10
(venv) PS C:\Users\userd\Downloads\flaskbb-master\flaskbb-master> Get-Content requirements.txt | Where-Object { $_ -match "^(Flask|Werkzeug|SQLAlchemy|Jinja2|Celery)" }
celery==5.4.0
Flask==3.0.3
Flask-Alembic==3.0.1
flask-allows @ git+https://github.com/flaskbb/flask-allows.git@master#egg=flask-allows
Flask-BabelPlus==2.2.0
Flask-Caching==2.1.0
Flask-DebugToolbar==0.14.1
flask-debugtoolbar-warnings==0.1.0
Flask-Limiter==3.6.0
Flask-Login==0.6.3
Flask-Mail==0.9.1
flask-redis==0.4.0
Flask-SQLAlchemy==3.1.1
Flask-Themes2==1.0.1
flask-whooshee==0.9.1
Flask-WTF==1.2.1
Jinja2==3.1.3
SQLAlchemy==2.0.29
SQLAlchemy-Utils==0.41.2
Werkzeug==3.0.2
(venv) PS C:\Users\userd\Downloads\flaskbb-master\flaskbb-master> python --version
Python 3.13.0
(venv) PS C:\Users\userd\Downloads\flaskbb-master\flaskbb-master>
15°C
Search
ENG
8:29 PM
11/2/2025


Windows PowerShell
(venv) PS C:\Users\userd\Downloads\flaskbb-master\flaskbb-master> cat requirements.txt | Select-String -Pattern "(Flask|Werkzeug|SQLAlchemy|Jinja2|Celery)"
celery==5.4.0
Flask==3.0.3
Flask-Alembic==3.0.1
flask-allows @
git+https://github.com/flaskbb/flask-allows.git@master#egg=flask-allows
Flask-BabelPlus==2.2.0
Flask-Caching==2.1.0
Flask-DebugToolbar==0.14.1
flask-debugtoolbar-warnings==0.1.0
Flask-Limiter==3.6.0
Flask-Login==0.6.3
Flask-Mail==0.9.1
flask-redis==0.4.0
Flask-SQLAlchemy==3.1.1
Flask-Themes2==1.0.1
flask-whooshee==0.9.1
Flask-WTF==1.2.1
Jinja2==3.1.3
SQLAlchemy==2.0.29
SQLAlchemy-Utils==0.41.2
Werkzeug==3.0.2
WTForms-SQLAlchemy==0.4.1
(venv) PS C:\Users\userd\Downloads\flaskbb-master\flaskbb-master> cat outdated_packages.txt
Package Version Latest Type
psutil 7.1.2 7.1.3 wheel
safety-schemas 0.0.16 0.0.17 wheel
(venv) PS C:\Users\userd\Downloads\flaskbb-master\flaskbb-master> pipdeptree | Select-String -Pattern "(Flask|Werkzeug|SQLAlchemy)" | Select-Object -First 10
(venv) PS C:\Users\userd\Downloads\flaskbb-master\flaskbb-master> Get-Content requirements.txt | Where-Object { $_ -match "^(Flask|Werkzeug|SQLAlchemy|Jinja2|Celery)" }
celery==5.4.0
Flask==3.0.3
Flask-Alembic==3.0.1
flask-allows @ git+https://github.com/flaskbb/flask-allows.git@master#egg=flask-allows
Flask-BabelPlus==2.2.0

5. Preliminary Maintenance Priorities

1. **Refactor Complex and Coupled Modules**
   Some files, like flaskbb/forum/views.py and flaskbb/message/views.py, have complicated and deeply nested code. They are also linked to bugs (#639, #675). Splitting them into smaller, clearer functions will make the code easier to understand, test, and maintain.

2. **Improve Documentation and Plugin Setup Guide**
   Issue #685 and Pylint reports show missing or unclear documentation. Many functions and plugins lack explanations. Adding docstrings, comments, and setup instructions will help new developers understand and use the system more easily.

3. **Fix Circular Dependencies and Code Duplication**
   Modules such as utils/helpers.py and forum/views.py have circular imports and repeated code (issue #677). Cleaning this up will make the project more organized and reduce future errors.

4. **Update Python and Package Versions**
   Several issues (#682, #666, #659) show that FlaskBB has old or incompatible dependencies for Python 3.13. Updating all libraries, especially safety-schemas (0.0.16 → 0.0.17), and testing the app after upgrades will keep it stable and secure.

5. **Increase Test Coverage and Automate Testing**
   Some areas, like the message/ and forum/ modules, lack enough tests (issue #661). Adding more unit and integration tests with pytest, and automating them in CI/CD, will catch bugs early and improve reliability.

6. **Improve Security and Use Modern Tools**
   The old safety check command is deprecated, and issue #680 reports an XSS vulnerability. Replacing it with a safety scan, fixing security flaws, and running regular checks will protect the application from attacks.

7. **Enhance User Interface and Responsiveness**
   Issues (#650, #625, #601, #671) show UI problems like layout errors and poor mobile support. Fixing CSS and making templates responsive will improve usability and the overall user experience.