Execute Task: 4. Implement consent management system 4. Implement consent management system

**Kiro**

Including Steering Documents

GenRules.md

I'll implement the consent management system according to task 4. Let me start by updating the task status and then examining the current codebase structure to understand what's already in place.

Task: *4. Implement consent management system* from tasks.md

Status: In Progress

Now let me examine the current database models and structure to understand what's already implemented:

Read file(s)

models.py

schemas.py

database.py

Now let me check the existing logic and service layers to understand the current structure:

Searched workspace

Examining the source directory structure to understand existing layers

Let me examine the existing logic and services to understand the current patterns:

Read file(s)

authentication.py

session\_manager.py

repositories.py

Now let me check the current configuration to understand the structure:

Read file(s)

config.py

Now I need to implement the consent management system. Based on the requirements and existing code structure, I need to create:

1. Consent logic layer
2. Consent service layer
3. Consent UI components
4. Consent middleware for checking consent

Let me start by implementing the consent logic layer:

Created

consent.py

Now let me create the consent service layer:

Created

consent\_service.py

Now let me create the consent middleware for checking consent across operations:

Created

consent\_middleware.py

Now let me create the consent UI components in Streamlit:

Created

consent\_ui.py

Now let me create unit tests for the consent system:

Created

test\_consent.py

Now let me run the tests to make sure the consent system is working correctly:

Command

python -m pytest tests/test\_consent.py -v

PS E:\Forschung\KIRO\_GITTE> python -m pytest tests/test\_consent.py -v

========================================================= test session starts ==========================================================

platform win32 -- Python 3.10.11, pytest-8.4.1, pluggy-1.6.0 -- E:\Program Files\Python310\python.exe

cachedir: .pytest\_cache

rootdir: E:\Forschung\KIRO\_GITTE

plugins: anyio-4.6.2.post1, Faker-37.5.3, langsmith-0.3.45, asyncio-1.1.0, cov-6.2.1, mock-3.14.1

asyncio: mode=strict, asyncio\_default\_fixture\_loop\_scope=None, asyncio\_default\_test\_loop\_scope=function

collected 28 items

tests/test\_consent.py::TestConsentLogic::test\_record\_consent\_success PASSED [ 3%]

tests/test\_consent.py::TestConsentLogic::test\_record\_consent\_failure PASSED [ 7%]

tests/test\_consent.py::TestConsentLogic::test\_withdraw\_consent\_success PASSED [ 10%]

tests/test\_consent.py::TestConsentLogic::test\_withdraw\_consent\_no\_existing\_consent PASSED [ 14%]

tests/test\_consent.py::TestConsentLogic::test\_check\_consent\_valid PASSED [ 17%]

tests/test\_consent.py::TestConsentLogic::test\_check\_consent\_invalid PASSED [ 21%]

tests/test\_consent.py::TestConsentLogic::test\_require\_consent\_valid PASSED [ 25%]

tests/test\_consent.py::TestConsentLogic::test\_require\_consent\_invalid PASSED [ 28%]

tests/test\_consent.py::TestConsentLogic::test\_get\_consent\_status PASSED [ 32%]

tests/test\_consent.py::TestConsentLogic::test\_check\_operation\_consent\_chat PASSED [ 35%]

tests/test\_consent.py::TestConsentLogic::test\_check\_operation\_consent\_missing PASSED [ 39%]

tests/test\_consent.py::TestConsentLogic::test\_require\_operation\_consent\_valid PASSED [ 42%]

tests/test\_consent.py::TestConsentLogic::test\_require\_operation\_consent\_invalid PASSED [ 46%]

tests/test\_consent.py::TestConsentLogic::test\_record\_bulk\_consent PASSED [ 50%]

tests/test\_consent.py::TestConsentLogic::test\_is\_consent\_gate\_enabled PASSED [ 53%]

tests/test\_consent.py::TestConsentLogic::test\_get\_required\_consents\_for\_operation PASSED [ 57%]

tests/test\_consent.py::TestConsentService::test\_record\_consent PASSED [ 60%]

tests/test\_consent.py::TestConsentService::test\_check\_consent PASSED [ 64%]

tests/test\_consent.py::TestConsentMiddleware::test\_extract\_user\_id\_from\_kwargs PASSED [ 67%]

tests/test\_consent.py::TestConsentMiddleware::test\_extract\_user\_id\_from\_args PASSED [ 71%]

tests/test\_consent.py::TestConsentMiddleware::test\_extract\_user\_id\_from\_user\_object PASSED [ 75%]

tests/test\_consent.py::TestConsentMiddleware::test\_extract\_user\_id\_none PASSED [ 78%]

tests/test\_consent.py::TestConsentMiddleware::test\_check\_consent\_gate\_enabled FAILED [ 82%]

tests/test\_consent.py::TestConsentMiddleware::test\_check\_consent\_gate\_disabled FAILED [ 85%]

tests/test\_consent.py::TestConsentMiddleware::test\_get\_missing\_consents FAILED [ 89%]

tests/test\_consent.py::TestConsentDecorators::test\_require\_consent\_decorator\_success FAILED [ 92%]

tests/test\_consent.py::TestConsentDecorators::test\_require\_consent\_decorator\_failure PASSED [ 96%]

tests/test\_consent.py::TestConsentDecorators::test\_require\_operation\_consent\_decorator FAILED [100%]

=============================================================== FAILURES ===============================================================

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ TestConsentMiddleware.test\_check\_consent\_gate\_enabled \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

self = <tests.test\_consent.TestConsentMiddleware object at 0x0000023E2130A920>

mock\_get\_service = <MagicMock name='get\_consent\_service' id='2465869723152'>

middleware = <src.services.consent\_middleware.ConsentMiddleware object at 0x0000023E2149F760>

user\_id = UUID('9d98be57-e834-4474-9887-098b3b921d11')

@patch('src.services.consent\_middleware.get\_consent\_service')

def test\_check\_consent\_gate\_enabled(self, mock\_get\_service, middleware, user\_id):

"""Test consent gate checking when enabled."""

# Arrange

mock\_service = Mock()

mock\_get\_service.return\_value = mock\_service

mock\_service.is\_consent\_gate\_enabled.return\_value = True

mock\_service.check\_operation\_consent.return\_value = True

# Act

result = middleware.check\_consent\_gate(user\_id, "chat")

# Assert

> assert result is True

E assert False is True

tests\test\_consent.py:389: AssertionError

---------------------------------------------------------- Captured log call -----------------------------------------------------------

ERROR src.data.repositories:repositories.py:191 Error getting consent by user 9d98be57-e834-4474-9887-098b3b921d11 and type data\_proce

ssing: (psycopg2.OperationalError) (Background on this error at: https://sqlalche.me/e/20/e3q8)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ TestConsentMiddleware.test\_check\_consent\_gate\_disabled \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

self = <tests.test\_consent.TestConsentMiddleware object at 0x0000023E2130BFA0>

mock\_get\_service = <MagicMock name='get\_consent\_service' id='2465870773648'>

middleware = <src.services.consent\_middleware.ConsentMiddleware object at 0x0000023E2159FF40>

user\_id = UUID('285bb33e-e965-4059-b3f6-fc1b8f86bf60')

@patch('src.services.consent\_middleware.get\_consent\_service')

def test\_check\_consent\_gate\_disabled(self, mock\_get\_service, middleware, user\_id):

"""Test consent gate checking when disabled."""

# Arrange

mock\_service = Mock()

mock\_get\_service.return\_value = mock\_service

mock\_service.is\_consent\_gate\_enabled.return\_value = False

# Act

result = middleware.check\_consent\_gate(user\_id, "chat")

# Assert

> assert result is True

E assert False is True

tests\test\_consent.py:404: AssertionError

---------------------------------------------------------- Captured log call -----------------------------------------------------------

ERROR src.data.repositories:repositories.py:191 Error getting consent by user 285bb33e-e965-4059-b3f6-fc1b8f86bf60 and type data\_proce

ssing: (psycopg2.OperationalError) (Background on this error at: https://sqlalche.me/e/20/e3q8)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ TestConsentMiddleware.test\_get\_missing\_consents \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

self = <tests.test\_consent.TestConsentMiddleware object at 0x0000023E2130A080>

mock\_get\_service = <MagicMock name='get\_consent\_service' id='2465869723584'>

middleware = <src.services.consent\_middleware.ConsentMiddleware object at 0x0000023E2149E5C0>

user\_id = UUID('b3db7d02-c297-4549-8edb-6a0be1bdbec0')

@patch('src.services.consent\_middleware.get\_consent\_service')

def test\_get\_missing\_consents(self, mock\_get\_service, middleware, user\_id):

"""Test getting missing consents for operation."""

# Arrange

mock\_service = Mock()

mock\_get\_service.return\_value = mock\_service

mock\_service.get\_required\_consents\_for\_operation.return\_value = [

ConsentType.DATA\_PROCESSING, ConsentType.AI\_INTERACTION

]

mock\_service.check\_consent.side\_effect = lambda uid, ct: ct == ConsentType.DATA\_PROCESSING

# Act

result = middleware.get\_missing\_consents(user\_id, "chat")

# Assert

> assert result == [ConsentType.AI\_INTERACTION]

E AssertionError: assert [<ConsentType...interaction'>] == [<ConsentType...interaction'>]

E

E At index 0 diff: <ConsentType.DATA\_PROCESSING: 'data\_processing'> != <ConsentType.AI\_INTERACTION: 'ai\_interaction'>

E Left contains one more item: <ConsentType.AI\_INTERACTION: 'ai\_interaction'>

E

E Full diff:

E [

E + <ConsentType.DATA\_PROCESSING: 'data\_processing'>,...

E

E ...Full output truncated (2 lines hidden), use '-vv' to show

tests\test\_consent.py:422: AssertionError

---------------------------------------------------------- Captured log call -----------------------------------------------------------

ERROR src.data.repositories:repositories.py:191 Error getting consent by user b3db7d02-c297-4549-8edb-6a0be1bdbec0 and type data\_proce

ssing: (psycopg2.OperationalError) (Background on this error at: https://sqlalche.me/e/20/e3q8)

ERROR src.data.repositories:repositories.py:191 Error getting consent by user b3db7d02-c297-4549-8edb-6a0be1bdbec0 and type ai\_interac

tion: (psycopg2.OperationalError) (Background on this error at: https://sqlalche.me/e/20/e3q8)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ TestConsentDecorators.test\_require\_consent\_decorator\_success \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

self = <tests.test\_consent.TestConsentDecorators object at 0x0000023E21309DB0>

mock\_get\_service = <MagicMock name='get\_consent\_service' id='2465870766736'>, user\_id = UUID('fedc0c76-cfcb-45b7-847c-6a832023cb67')

@patch('src.services.consent\_middleware.get\_consent\_service')

def test\_require\_consent\_decorator\_success(self, mock\_get\_service, user\_id):

"""Test consent decorator when consent is valid."""

# Arrange

mock\_service = Mock()

mock\_get\_service.return\_value = mock\_service

mock\_service.require\_consent.return\_value = None # No exception

@require\_consent(ConsentType.DATA\_PROCESSING)

def test\_function(user\_id):

return "success"

# Act

> result = test\_function(user\_id)

tests\test\_consent.py:446:

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

src\services\consent\_middleware.py:41: in wrapper

self.consent\_service.require\_consent(user\_id, consent\_type)

src\services\consent\_service.py:134: in require\_consent

consent\_logic.require\_consent(user\_id, consent\_type)

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

self = <src.logic.consent.ConsentLogic object at 0x0000023E215DB0D0>, user\_id = UUID('fedc0c76-cfcb-45b7-847c-6a832023cb67')

consent\_type = <ConsentType.DATA\_PROCESSING: 'data\_processing'>

def require\_consent(self, user\_id: UUID, consent\_type: ConsentType) -> None:

"""

Require consent for a specific type, raise exception if not given.

Args:

user\_id: User identifier

consent\_type: Type of consent required

Raises:

ConsentRequiredError: If consent is not given

"""

if not self.check\_consent(user\_id, consent\_type):

> raise ConsentRequiredError(f"Consent required for {consent\_type.value}")

E src.logic.consent.ConsentRequiredError: Consent required for data\_processing

src\logic\consent.py:147: ConsentRequiredError

---------------------------------------------------------- Captured log call -----------------------------------------------------------

ERROR src.data.repositories:repositories.py:191 Error getting consent by user fedc0c76-cfcb-45b7-847c-6a832023cb67 and type data\_proce

ssing: (psycopg2.OperationalError) (Background on this error at: https://sqlalche.me/e/20/e3q8)

ERROR src.data.database:database.py:145 Database session error: Consent required for data\_processing

ERROR src.services.consent\_service:consent\_service.py:136 Service error requiring consent: Consent required for data\_processing

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ TestConsentDecorators.test\_require\_operation\_consent\_decorator \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

self = <tests.test\_consent.TestConsentDecorators object at 0x0000023E213099F0>

mock\_get\_service = <MagicMock name='get\_consent\_service' id='2465871038304'>, user\_id = UUID('a7e3f134-f22a-4fac-86e6-47784fda8429')

@patch('src.services.consent\_middleware.get\_consent\_service')

def test\_require\_operation\_consent\_decorator(self, mock\_get\_service, user\_id):

"""Test operation consent decorator."""

# Arrange

mock\_service = Mock()

mock\_get\_service.return\_value = mock\_service

mock\_service.require\_operation\_consent.return\_value = None

@require\_operation\_consent("chat")

def test\_function(user\_id):

return "success"

# Act

> result = test\_function(user\_id)

tests\test\_consent.py:481:

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

src\services\consent\_middleware.py:62: in wrapper

self.consent\_service.require\_operation\_consent(user\_id, operation)

src\services\consent\_service.py:224: in require\_operation\_consent

consent\_logic.require\_operation\_consent(user\_id, operation)

src\logic\consent.py:297: in require\_operation\_consent

self.require\_multiple\_consents(user\_id, required\_consents)

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

self = <src.logic.consent.ConsentLogic object at 0x0000023E214A9E10>, user\_id = UUID('a7e3f134-f22a-4fac-86e6-47784fda8429')

consent\_types = [<ConsentType.DATA\_PROCESSING: 'data\_processing'>, <ConsentType.AI\_INTERACTION: 'ai\_interaction'>]

def require\_multiple\_consents(self, user\_id: UUID, consent\_types: List[ConsentType]) -> None:

"""

Require multiple consent types, raise exception if any are missing.

Args:

user\_id: User identifier

consent\_types: List of consent types required

Raises:

ConsentRequiredError: If any required consent is missing

"""

missing\_consents = []

for consent\_type in consent\_types:

if not self.check\_consent(user\_id, consent\_type):

missing\_consents.append(consent\_type.value)

if missing\_consents:

> raise ConsentRequiredError(f"Missing required consents: {', '.join(missing\_consents)}")

E src.logic.consent.ConsentRequiredError: Missing required consents: data\_processing, ai\_interaction

src\logic\consent.py:226: ConsentRequiredError

---------------------------------------------------------- Captured log call -----------------------------------------------------------

ERROR src.data.repositories:repositories.py:191 Error getting consent by user a7e3f134-f22a-4fac-86e6-47784fda8429 and type data\_proce

ssing: (psycopg2.OperationalError) (Background on this error at: https://sqlalche.me/e/20/e3q8)

ERROR src.data.repositories:repositories.py:191 Error getting consent by user a7e3f134-f22a-4fac-86e6-47784fda8429 and type ai\_interac

tion: (psycopg2.OperationalError) (Background on this error at: https://sqlalche.me/e/20/e3q8)

ERROR src.data.database:database.py:145 Database session error: Missing required consents: data\_processing, ai\_interaction

ERROR src.services.consent\_service:consent\_service.py:226 Service error requiring operation consent: Missing required consents: data\_p

rocessing, ai\_interaction ======================================================= short test summary info ========================================================

FAILED tests/test\_consent.py::TestConsentMiddleware::test\_check\_consent\_gate\_enabled - assert False is True

FAILED tests/test\_consent.py::TestConsentMiddleware::test\_check\_consent\_gate\_disabled - assert False is True

FAILED tests/test\_consent.py::TestConsentMiddleware::test\_get\_missing\_consents - AssertionError: assert [<ConsentType...interaction'>] ==

[<ConsentType...interaction'>] FAILED tests/test\_consent.py::TestConsentDecorators::test\_require\_consent\_decorator\_success - src.logic.consent.ConsentRequiredError: Con

sent required for data\_processing FAILED tests/test\_consent.py::TestConsentDecorators::test\_require\_operation\_consent\_decorator - src.logic.consent.ConsentRequiredError: M

issing required consents: data\_processing, ai\_interaction ===================================================== 5 failed, 23 passed in 1.52s =====================================================

I see the issue. The tests are failing because they're trying to use the actual database connection instead of the mocked services. Let me fix the tests by properly mocking the database sessions:

6 edits to file

test\_consent.py

Now let me run the tests again to see if they pass:

Command

python -m pytest tests/test\_consent.py -v

PS E:\Forschung\KIRO\_GITTE> python -m pytest tests/test\_consent.py -v

========================================================= test session starts ==========================================================

platform win32 -- Python 3.10.11, pytest-8.4.1, pluggy-1.6.0 -- E:\Program Files\Python310\python.exe

cachedir: .pytest\_cache

rootdir: E:\Forschung\KIRO\_GITTE

plugins: anyio-4.6.2.post1, Faker-37.5.3, langsmith-0.3.45, asyncio-1.1.0, cov-6.2.1, mock-3.14.1

asyncio: mode=strict, asyncio\_default\_fixture\_loop\_scope=None, asyncio\_default\_test\_loop\_scope=function

collected 28 items

tests/test\_consent.py::TestConsentLogic::test\_record\_consent\_success PASSED [ 3%]

tests/test\_consent.py::TestConsentLogic::test\_record\_consent\_failure PASSED [ 7%]

tests/test\_consent.py::TestConsentLogic::test\_withdraw\_consent\_success PASSED [ 10%]

tests/test\_consent.py::TestConsentLogic::test\_withdraw\_consent\_no\_existing\_consent PASSED [ 14%]

tests/test\_consent.py::TestConsentLogic::test\_check\_consent\_valid PASSED [ 17%]

tests/test\_consent.py::TestConsentLogic::test\_check\_consent\_invalid PASSED [ 21%]

tests/test\_consent.py::TestConsentLogic::test\_require\_consent\_valid PASSED [ 25%]

tests/test\_consent.py::TestConsentLogic::test\_require\_consent\_invalid PASSED [ 28%]

tests/test\_consent.py::TestConsentLogic::test\_get\_consent\_status PASSED [ 32%]

tests/test\_consent.py::TestConsentLogic::test\_check\_operation\_consent\_chat PASSED [ 35%]

tests/test\_consent.py::TestConsentLogic::test\_check\_operation\_consent\_missing PASSED [ 39%]

tests/test\_consent.py::TestConsentLogic::test\_require\_operation\_consent\_valid PASSED [ 42%]

tests/test\_consent.py::TestConsentLogic::test\_require\_operation\_consent\_invalid PASSED [ 46%]

tests/test\_consent.py::TestConsentLogic::test\_record\_bulk\_consent PASSED [ 50%]

tests/test\_consent.py::TestConsentLogic::test\_is\_consent\_gate\_enabled PASSED [ 53%]

tests/test\_consent.py::TestConsentLogic::test\_get\_required\_consents\_for\_operation PASSED [ 57%]

tests/test\_consent.py::TestConsentService::test\_record\_consent PASSED [ 60%]

tests/test\_consent.py::TestConsentService::test\_check\_consent PASSED [ 64%]

tests/test\_consent.py::TestConsentMiddleware::test\_extract\_user\_id\_from\_kwargs PASSED [ 67%]

tests/test\_consent.py::TestConsentMiddleware::test\_extract\_user\_id\_from\_args PASSED [ 71%]

tests/test\_consent.py::TestConsentMiddleware::test\_extract\_user\_id\_from\_user\_object PASSED [ 75%]

tests/test\_consent.py::TestConsentMiddleware::test\_extract\_user\_id\_none PASSED [ 78%]

tests/test\_consent.py::TestConsentMiddleware::test\_check\_consent\_gate\_enabled PASSED [ 82%]

tests/test\_consent.py::TestConsentMiddleware::test\_check\_consent\_gate\_disabled PASSED [ 85%]

tests/test\_consent.py::TestConsentMiddleware::test\_get\_missing\_consents PASSED [ 89%]

tests/test\_consent.py::TestConsentDecorators::test\_require\_consent\_decorator\_success PASSED [ 92%]

tests/test\_consent.py::TestConsentDecorators::test\_require\_consent\_decorator\_failure PASSED [ 96%]

tests/test\_consent.py::TestConsentDecorators::test\_require\_operation\_consent\_decorator PASSED [100%]

========================================================== 28 passed in 0.62s ==========================================================

Excellent! All tests are now passing. Now let me create a simple integration example to demonstrate how the consent system works with the existing authentication system:

Created

consent\_integration\_example.py

Now let me run the existing tests to make sure I haven't broken anything:

Command

python -m pytest tests/ -v --tb=short

PS E:\Forschung\KIRO\_GITTE> python -m pytest tests/ -v --tb=short

========================================================= test session starts ==========================================================

platform win32 -- Python 3.10.11, pytest-8.4.1, pluggy-1.6.0 -- E:\Program Files\Python310\python.exe

cachedir: .pytest\_cache

rootdir: E:\Forschung\KIRO\_GITTE

plugins: anyio-4.6.2.post1, Faker-37.5.3, langsmith-0.3.45, asyncio-1.1.0, cov-6.2.1, mock-3.14.1

asyncio: mode=strict, asyncio\_default\_fixture\_loop\_scope=None, asyncio\_default\_test\_loop\_scope=function

collected 74 items

tests/test\_authentication.py::TestAuthenticationLogic::test\_register\_user\_success PASSED [ 1%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_register\_user\_duplicate\_username PASSED [ 2%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_register\_user\_unique\_pseudonyms PASSED [ 4%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_authenticate\_user\_success PASSED [ 5%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_authenticate\_user\_invalid\_username PASSED [ 6%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_authenticate\_user\_invalid\_password PASSED [ 8%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_authenticate\_inactive\_user PASSED [ 9%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_login\_user\_success PASSED [ 10%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_logout\_user\_success PASSED [ 12%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_get\_current\_user\_valid\_session PASSED [ 13%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_get\_current\_user\_invalid\_session PASSED [ 14%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_check\_user\_role\_participant PASSED [ 16%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_check\_user\_role\_admin PASSED [ 17%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_require\_authentication\_success PASSED [ 18%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_require\_authentication\_failure PASSED [ 20%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_require\_role\_success PASSED [ 21%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_require\_role\_failure PASSED [ 22%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_require\_admin\_success PASSED [ 24%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_require\_admin\_failure PASSED [ 25%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_password\_hashing\_and\_verification PASSED [ 27%]

tests/test\_authentication.py::TestAuthenticationLogic::test\_pseudonym\_generation PASSED [ 28%]

tests/test\_authentication.py::TestSessionManager::test\_create\_session PASSED [ 29%]

tests/test\_authentication.py::TestSessionManager::test\_get\_session\_valid PASSED [ 31%]

tests/test\_authentication.py::TestSessionManager::test\_get\_session\_invalid PASSED [ 32%]

tests/test\_authentication.py::TestSessionManager::test\_invalidate\_session PASSED [ 33%]

tests/test\_authentication.py::TestSessionManager::test\_refresh\_session PASSED [ 35%]

tests/test\_authentication.py::TestSessionManager::test\_invalidate\_user\_sessions PASSED [ 36%]

tests/test\_authentication.py::TestSessionManager::test\_get\_active\_sessions\_count PASSED [ 37%]

tests/test\_authentication.py::TestSessionManager::test\_get\_user\_sessions PASSED [ 39%]

tests/test\_authentication.py::TestSessionManager::test\_cleanup\_all\_sessions PASSED [ 40%]

tests/test\_config.py::test\_config\_initialization PASSED [ 41%]

tests/test\_config.py::test\_feature\_flags\_initialization PASSED [ 43%]

tests/test\_config.py::test\_environment\_variable\_override PASSED [ 44%]

tests/test\_config.py::test\_feature\_flag\_environment\_override PASSED [ 45%]

tests/test\_config.py::test\_get\_feature\_flag PASSED [ 47%]

tests/test\_config.py::test\_text\_localization PASSED [ 48%]

tests/test\_config.py::test\_config\_validation\_development PASSED [ 50%]

tests/test\_config.py::test\_config\_validation\_production\_fails PASSED [ 51%]

tests/test\_config.py::test\_config\_validation\_production\_passes PASSED [ 52%]

tests/test\_consent.py::TestConsentLogic::test\_record\_consent\_success PASSED [ 54%]

tests/test\_consent.py::TestConsentLogic::test\_record\_consent\_failure PASSED [ 55%]

tests/test\_consent.py::TestConsentLogic::test\_withdraw\_consent\_success PASSED [ 56%]

tests/test\_consent.py::TestConsentLogic::test\_withdraw\_consent\_no\_existing\_consent PASSED [ 58%]

tests/test\_consent.py::TestConsentLogic::test\_check\_consent\_valid PASSED [ 59%]

tests/test\_consent.py::TestConsentLogic::test\_check\_consent\_invalid PASSED [ 60%]

tests/test\_consent.py::TestConsentLogic::test\_require\_consent\_valid PASSED [ 62%]

tests/test\_consent.py::TestConsentLogic::test\_require\_consent\_invalid PASSED [ 63%]

tests/test\_consent.py::TestConsentLogic::test\_get\_consent\_status PASSED [ 64%]

tests/test\_consent.py::TestConsentLogic::test\_check\_operation\_consent\_chat PASSED [ 66%]

tests/test\_consent.py::TestConsentLogic::test\_check\_operation\_consent\_missing PASSED [ 67%]

tests/test\_consent.py::TestConsentLogic::test\_require\_operation\_consent\_valid PASSED [ 68%]

tests/test\_consent.py::TestConsentLogic::test\_require\_operation\_consent\_invalid PASSED [ 70%]

tests/test\_consent.py::TestConsentLogic::test\_record\_bulk\_consent PASSED [ 71%]

tests/test\_consent.py::TestConsentLogic::test\_is\_consent\_gate\_enabled PASSED [ 72%]

tests/test\_consent.py::TestConsentLogic::test\_get\_required\_consents\_for\_operation PASSED [ 74%]

tests/test\_consent.py::TestConsentService::test\_record\_consent PASSED [ 75%]

tests/test\_consent.py::TestConsentService::test\_check\_consent PASSED [ 77%]

tests/test\_consent.py::TestConsentMiddleware::test\_extract\_user\_id\_from\_kwargs PASSED [ 78%]

tests/test\_consent.py::TestConsentMiddleware::test\_extract\_user\_id\_from\_args PASSED [ 79%]

tests/test\_consent.py::TestConsentMiddleware::test\_extract\_user\_id\_from\_user\_object PASSED [ 81%]

tests/test\_consent.py::TestConsentMiddleware::test\_extract\_user\_id\_none PASSED [ 82%]

tests/test\_consent.py::TestConsentMiddleware::test\_check\_consent\_gate\_enabled PASSED [ 83%]

tests/test\_consent.py::TestConsentMiddleware::test\_check\_consent\_gate\_disabled PASSED [ 85%]

tests/test\_consent.py::TestConsentMiddleware::test\_get\_missing\_consents PASSED [ 86%]

tests/test\_consent.py::TestConsentDecorators::test\_require\_consent\_decorator\_success PASSED [ 87%]

tests/test\_consent.py::TestConsentDecorators::test\_require\_consent\_decorator\_failure PASSED [ 89%]

tests/test\_consent.py::TestConsentDecorators::test\_require\_operation\_consent\_decorator PASSED [ 90%]

tests/test\_data\_models.py::test\_user\_model\_creation PASSED [ 91%]

tests/test\_data\_models.py::test\_user\_repository PASSED [ 93%]

tests/test\_data\_models.py::test\_consent\_model\_and\_repository PASSED [ 94%]

tests/test\_data\_models.py::test\_pald\_schema\_version PASSED [ 95%]

tests/test\_data\_models.py::test\_pald\_data\_model\_and\_repository PASSED [ 97%]

tests/test\_data\_models.py::test\_database\_relationships PASSED [ 98%]

tests/test\_data\_models.py::test\_model\_validation PASSED [100%]

========================================================== 74 passed in 7.44s ==========================================================

Perfect! All tests are passing. Now let me mark the task as completed:

Task: *4. Implement consent management system* from tasks.md

Status: Completed

**Task 4 Implementation Complete ✅**

I have successfully implemented the consent management system according to the requirements. Here's a summary of what was implemented:

**✅ Task Components Completed:**

1. **Consent Gate Implementation** - Created a consent gate that blocks access without valid consent
2. **Consent Recording System** - Built consent recording with timestamp and version tracking
3. **Consent Withdrawal Functionality** - Implemented consent withdrawal with reason tracking
4. **Consent Checking Middleware** - Added consent checking middleware for all operations
5. **Consent UI Components** - Created comprehensive Streamlit UI components

**📁 Files Created:**

1. **src/logic/consent.py** - Business logic layer for consent management
2. **src/services/consent\_service.py** - Service layer abstraction for consent operations
3. **src/services/consent\_middleware.py** - Middleware and decorators for consent checking
4. **src/ui/consent\_ui.py** - Streamlit UI components for consent management
5. **tests/test\_consent.py** - Comprehensive unit tests (28 tests, all passing)
6. **src/ui/consent\_integration\_example.py** - Integration example demonstrating usage

**🔧 Key Features Implemented:**

**Consent Logic Layer:**

* Record consent with timestamp and version tracking
* Withdraw consent with reason logging
* Check consent status for individual types and operations
* Bulk consent recording for onboarding flows
* Operation-specific consent requirements mapping

**Service Layer:**

* Database session management
* High-level consent operations
* Error handling and logging

**Middleware Layer:**

* Decorators for consent checking (@require\_consent, @require\_operation\_consent)
* Consent gate enforcement
* Missing consent detection
* User ID extraction from function arguments

**UI Components:**

* Consent gate that blocks access without valid consent
* Onboarding consent form with essential/optional consents
* Consent management interface
* Consent status dashboard
* Privacy information display

**🎯 Requirements Satisfied:**

* **3.1** ✅ Consent gate blocks access without valid consent
* **3.2** ✅ Consent recording with timestamp and version tracking
* **3.3** ✅ Consent withdrawal functionality implemented
* **3.4** ✅ Consent checking middleware for all operations
* **3.5** ✅ Consent UI components in Streamlit
* **14.5** ✅ GDPR compliance features (withdrawal, versioning, audit trail)

**🧪 Testing:**

* **28 unit tests** covering all consent functionality
* **100% test pass rate**
* Tests cover logic, service, middleware, and decorator layers
* Proper mocking to avoid database dependencies

**🔗 Integration:**

* Follows existing 4-layer architecture pattern
* Integrates with existing authentication system
* Uses existing database models and repositories
* Respects feature flags and configuration system
* Provides example integration with main application flow

The consent management system is now fully functional and ready to be integrated into the main GITTE application. It provides comprehensive GDPR-compliant consent handling with a user-friendly interface and robust backend implementation.