Mini Twitter

Design report

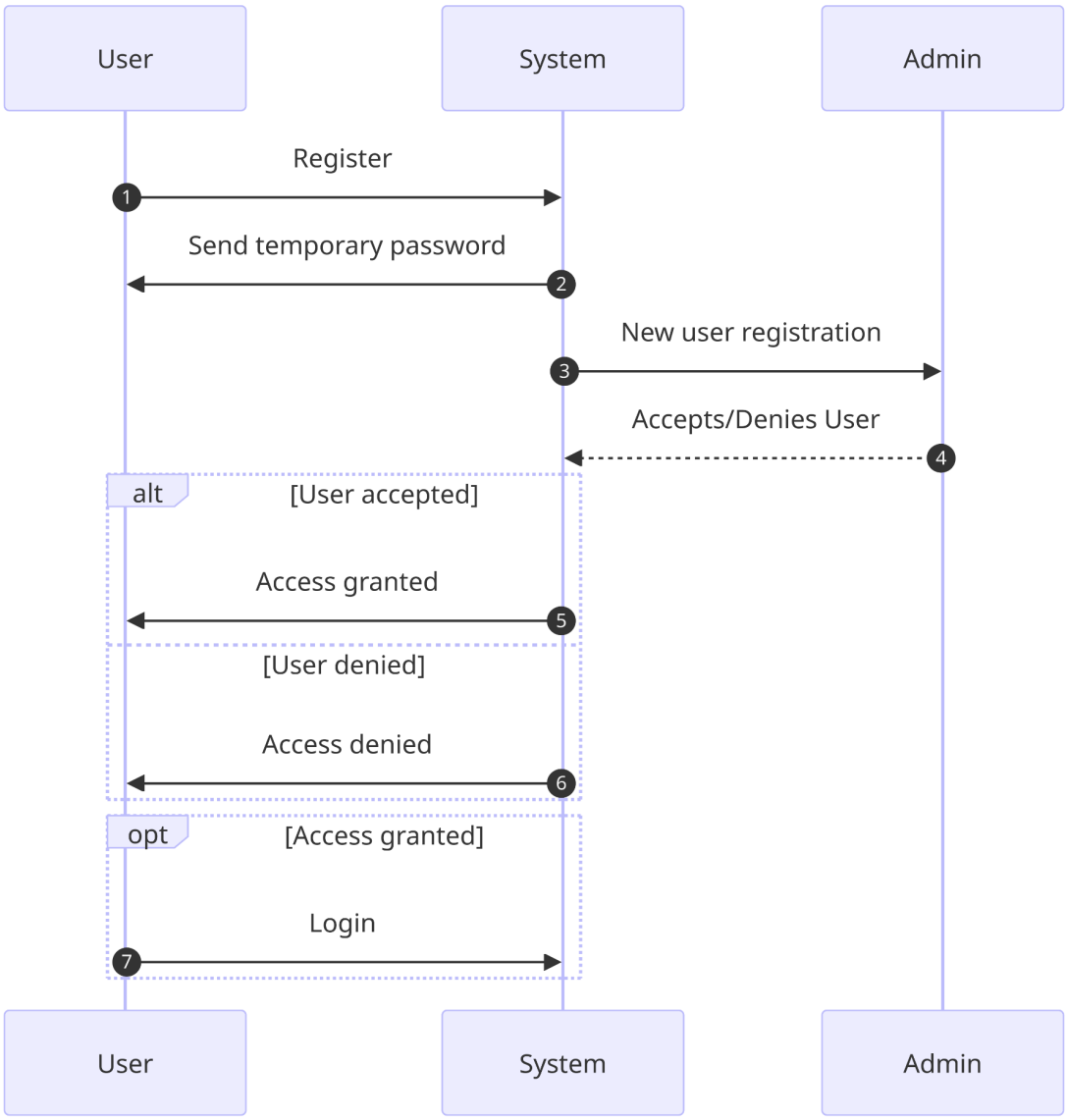
Chapters:

1. Overall picture of the system and use cases
2. Collaboration class and E-R Diagram
3. Detailed design and class diagrams
4. System screen prototypes
5. Group Memos
6. Github Link

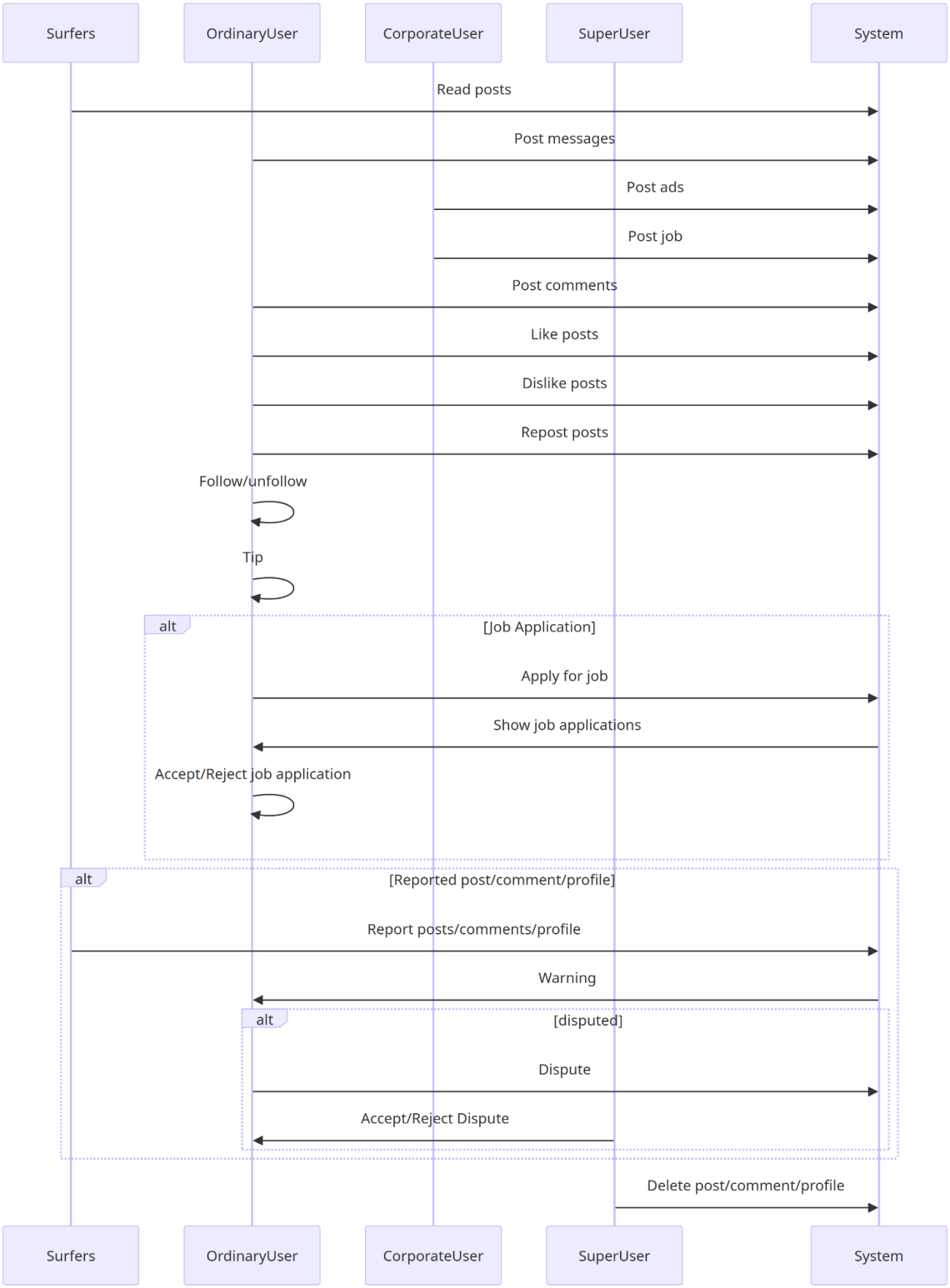
Group Members: Sarjeel Chowdhury, Fahim Hasan. Aquib Zaman

Overall picture of the system and use cases

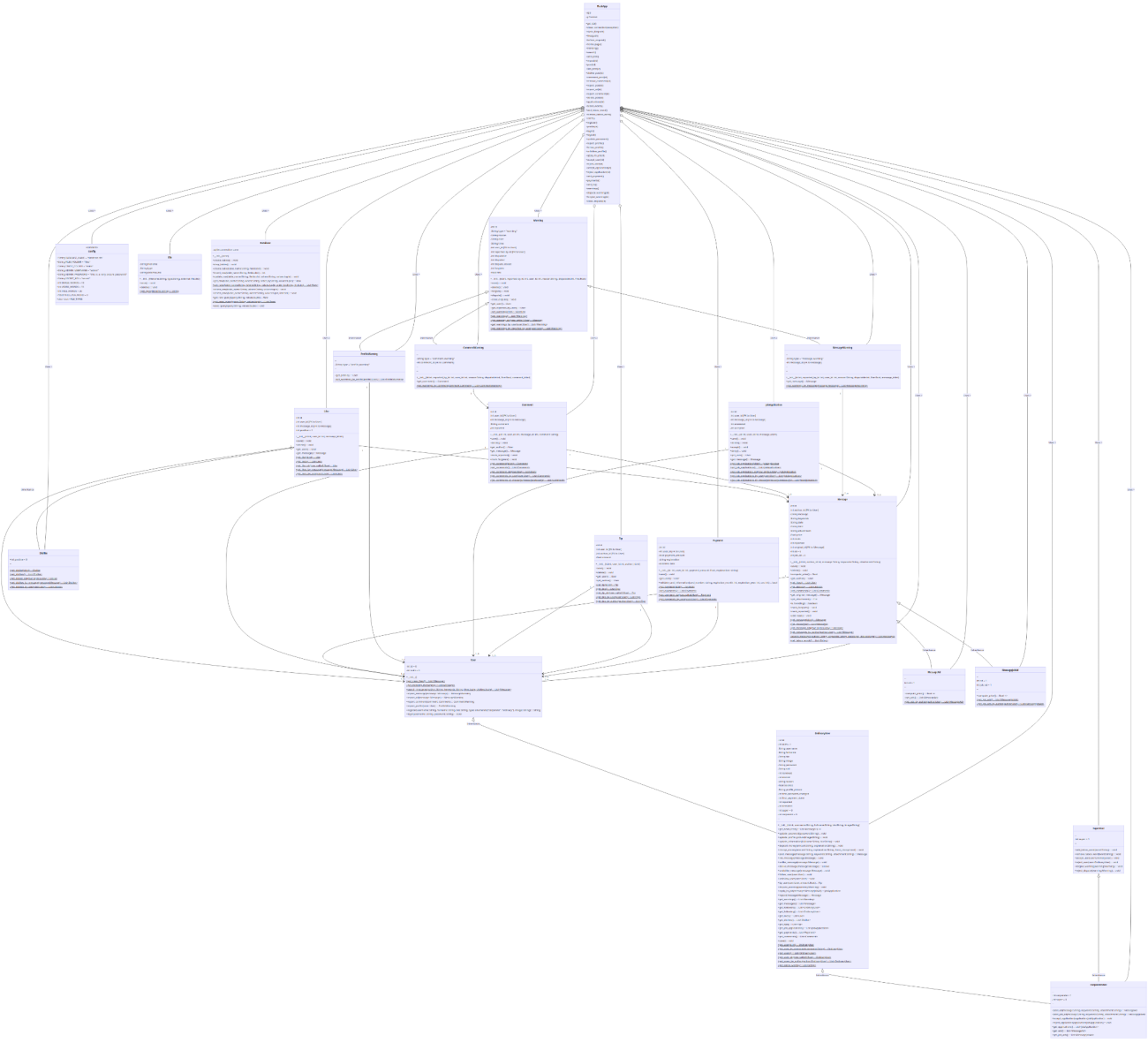
User registration/login functionality



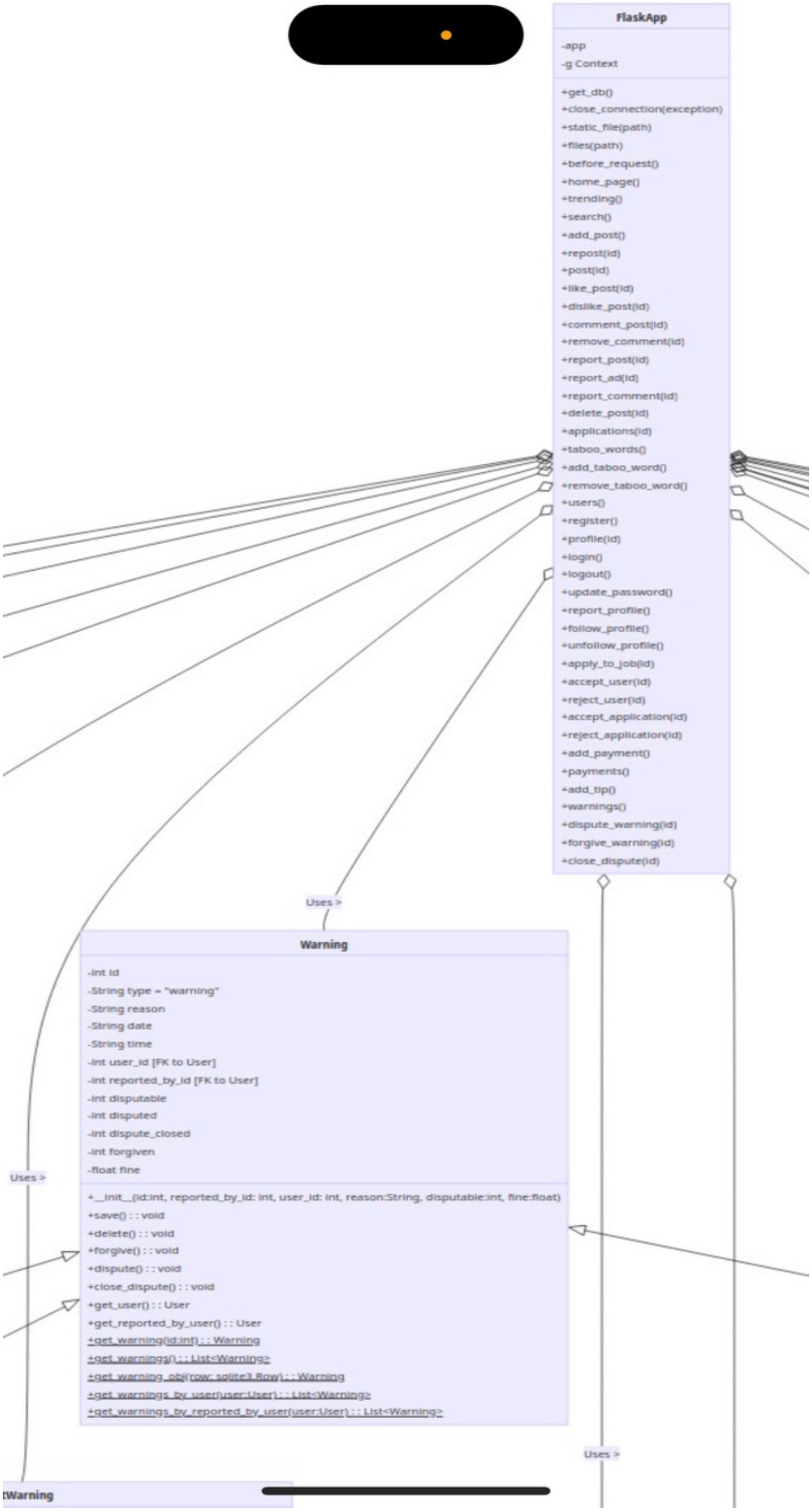
User actions in the system

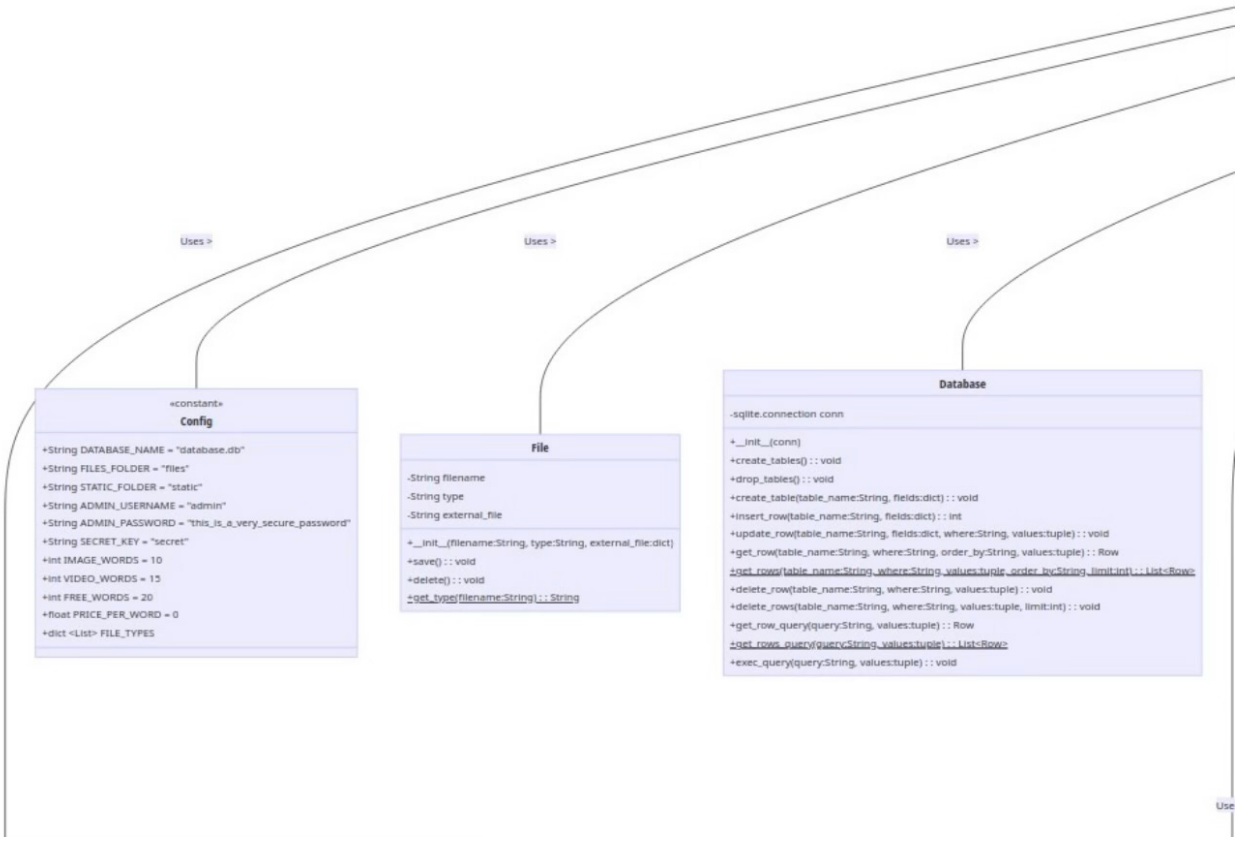


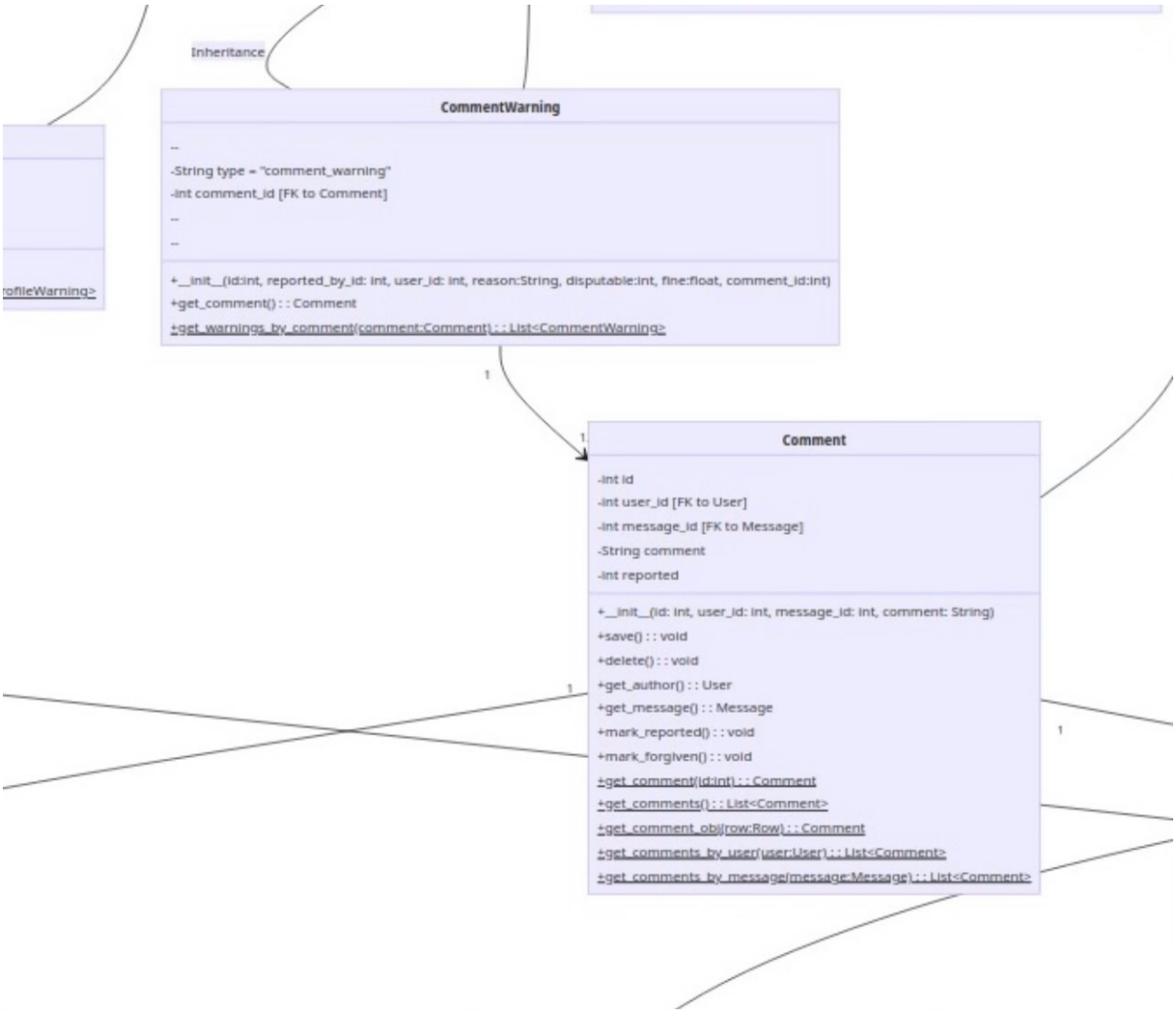
Collaboration class and E-R Diagram

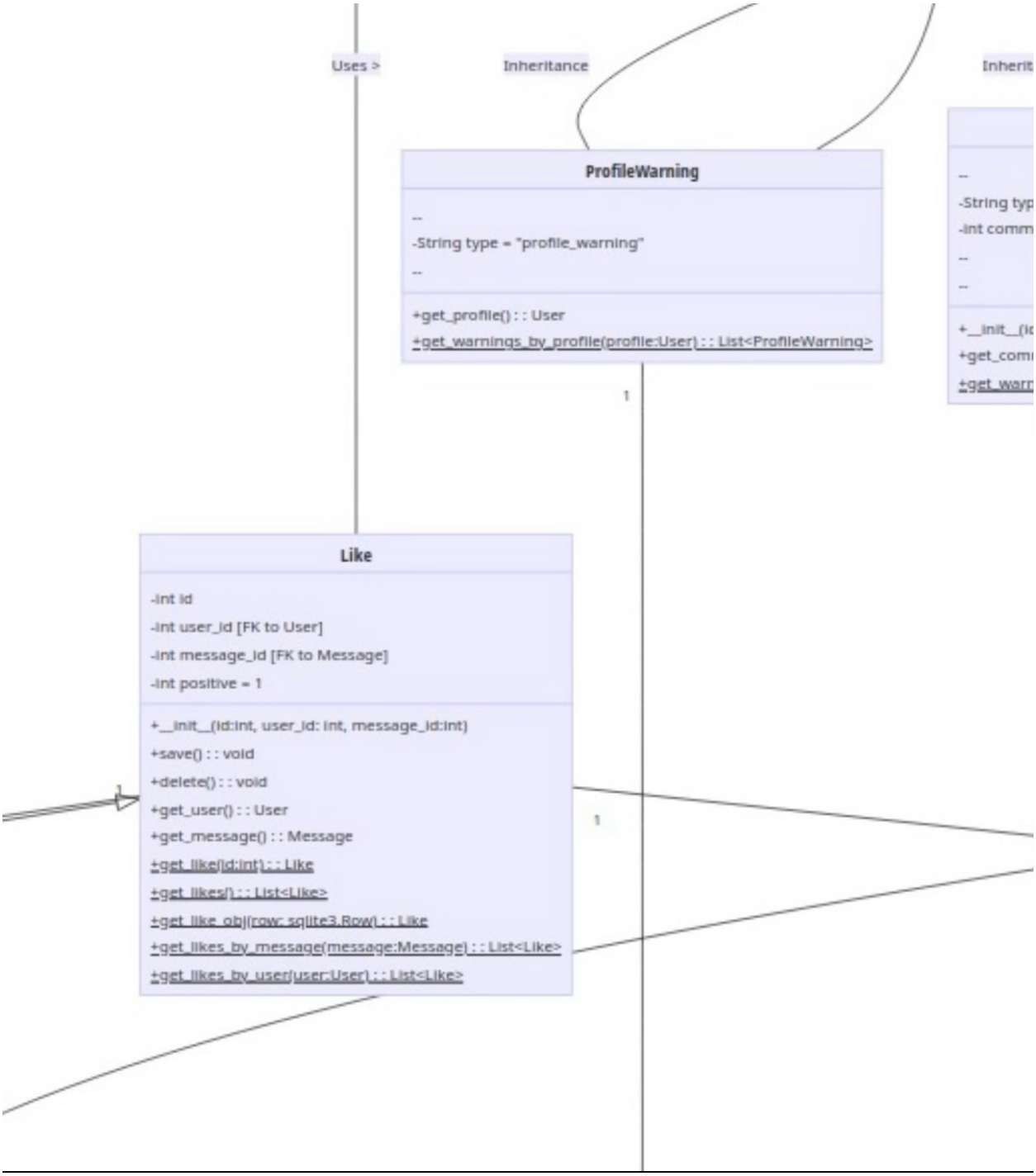


Screenshots of Collaboration diagram (zoomed in):

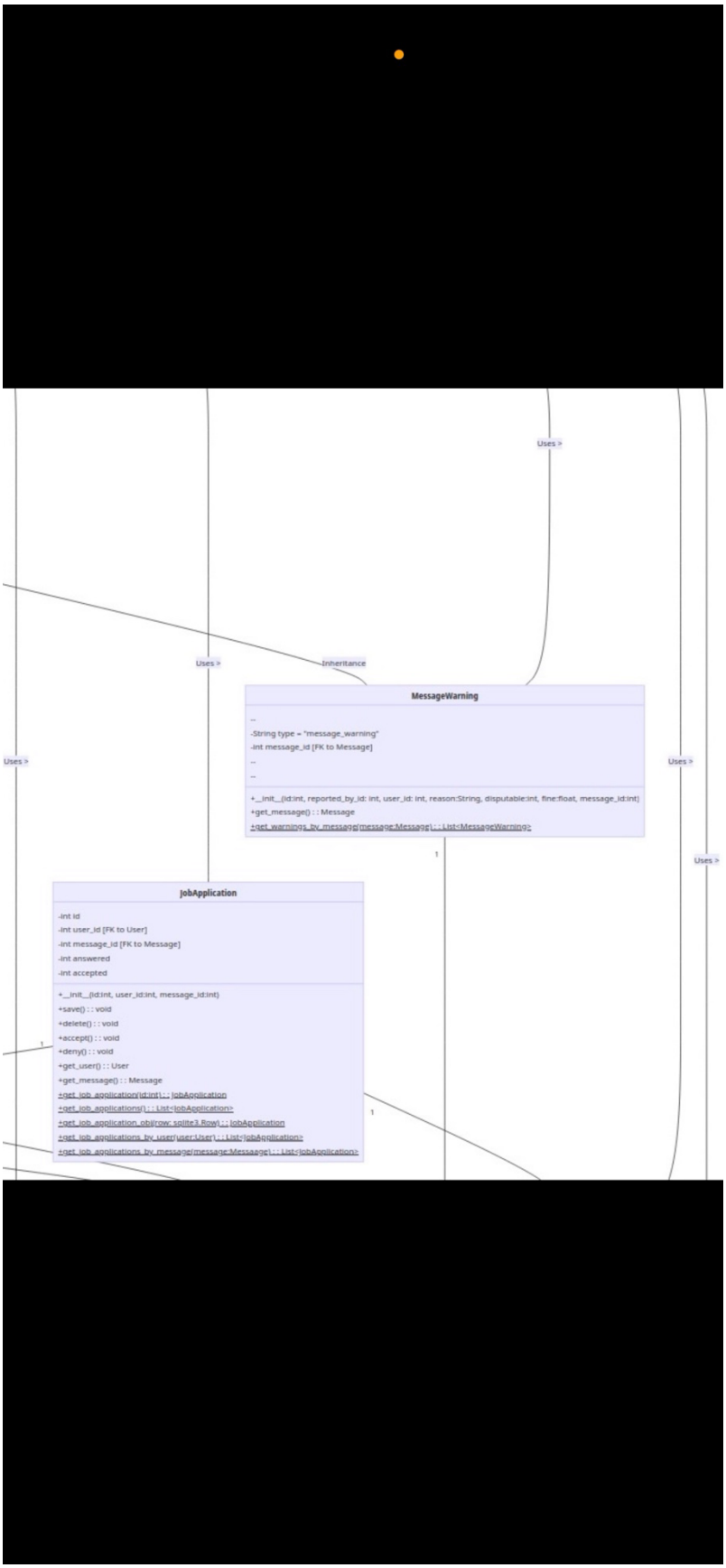


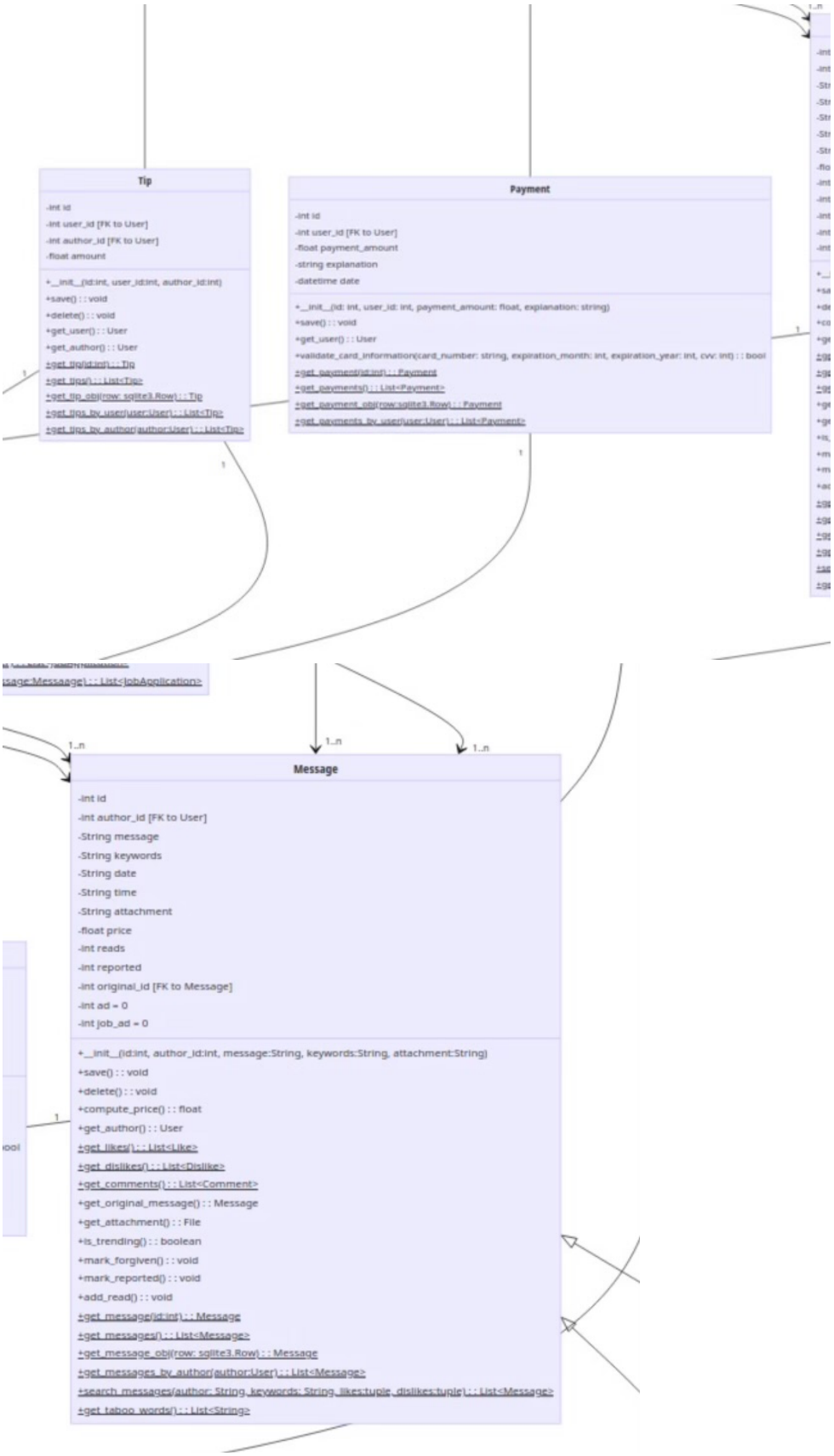

Uses >
Uses >
Uses >
«constant»
Config
+String DATABASE_NAME = "database.db"
+String FILES_FOLDER = "files"
+String STATIC_FOLDER = "static"
+String ADMIN_USERNAME = "admin"
+String ADMIN_PASSWORD = "this_is_a_very_secure_password"
+String SECRET_KEY = "secret"
+int IMAGE_WORDS = 10
+int VIDEO_WORDS = 15
+int FREE_WORDS = 20
+float PRICE_PER_WORD = 0
+dict <List> FILE_TYPES
File
-String filename
-String type
-String external_file
+__init__(filename:String, type:String, external_file:dict)
+save() : : void
+delete() : : void
+get_type(filename:String) : : String
Database
-sqlite.connection conn
+__init__(conn)
+create_tables() : : void
+drop_tables() : : void
+create_table(table_name:String, fields:dict) : : void
+insert_row(table_name:String, fields:dict) : : int
+update_row(table_name:String, fields:dict, where:String, values:tuple) : : void
+get_row(table_name:String, where:String, order_by:String, values:tuple) : : Row
+get_rows(table_name:String, where:String, values:tuple, order_by:String, limit:int) : : List<Row>
+delete_row(table_name:String, where:String, values:tuple) : : void
+delete_rows(table_name:String, where:String, values:tuple, limit:int) : : void
+get_row_query(query:String, values:tuple) : : Row
+get_rows_query(query:String, values:tuple) : : List<Row>
+exec_query(query:String, values:tuple) : : void

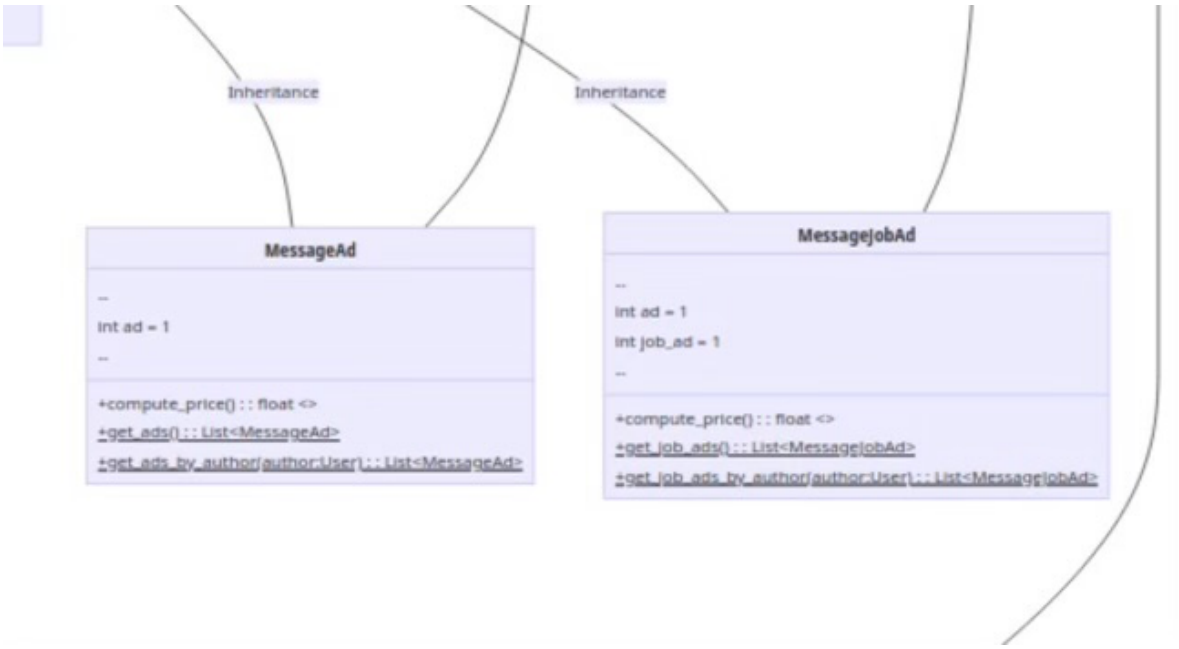

Inheritance
CommentWarning
--
-String type = "comment_warning"
-int comment_id [FK to Comment]
--
--
+__init__(id:int, reported_by_id: int, user_id: int, reason:String, disputable:int, fine:float, comment_id:int)
+get_comment() : : Comment
+get_warnings_by_comment(comment:Comment) : : List<CommentWarning>
ofileWarning>
1
1
Comment
-Int id
-Int user_id [FK to User]
-Int message_id [FK to Message]
-String comment
-Int reported
+__init__(id: int, user_id: int, message_id: int, comment: String)
+save() : : void
+delete() : : void
+get_author() : : User
+get_message() : : Message
+mark_reported() : : void
+mark_forgiven() : : void
+get_comment(id:int) : : Comment
+get_comments() : : List<Comment>
+get_comment_obj(row:Row) : : Comment
+get_comments_by_user(user:User) : : List<Comment>
+get_comments_by_message(message:Message) : : List<Comment>
1
1


Uses >
Inheritance
ProfileWarning
--
-String type = "profile_warning"
--
+get_profile() : : User
+get_warnings_by_profile(profile:User) : : List<ProfileWarning>
1
Like
-int id
-int user_id [FK to User]
-int message_id [FK to Message]
-int positive = 1
+__init__(id:int, user_id: int, message_id:int)
+save() : : void
+delete() : : void
+get_user() : : User
+get_message() : : Message
+get_like(id:int) : : Like
+get_likes() : : List<Like>
+get_like_obj(row: sqlite3.Row() : : Like
+get_likes_by_message(message:Message) : : List<Like>
+get_likes_by_user(user:User) : : List<Like>
1
1

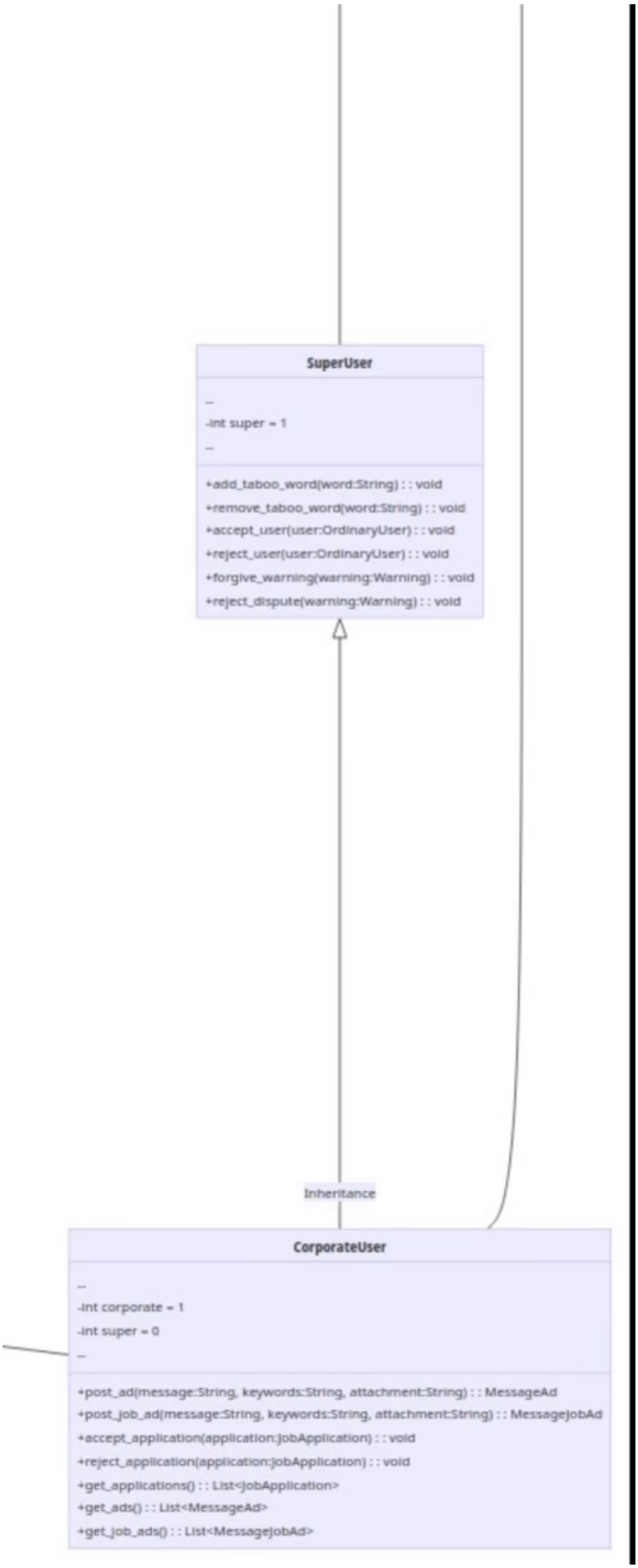

Inheritance
Dislike
+int positive = 0
--
+get_dislike(id:int) : : Dislike
+get_dislikes() : : List<Dislike>
+get_dislike_obj(row: sqlite3.Row) : : Dislike
+get_dislikes_by_message(message:Message) : : List<Dislike>
+get_dislikes_by_user(user:User) : : List<Dislike>
1..n
1..n
User
-int id = 0
-int auth = 0
+__init__()
+get_news_feed() : : List<Message>
+get_trending_messages() : : List<Message>
+search_messages(author: String, keywords: String, likes:tuple, dislikes:tuple) : : List<Message>
+report_message(message: Message) : : MessageWarning
+report_ad(message: Message) : : MessageWarning
+report_comment(comment: Comment) : : CommentWarning
+report_profile(user: User) : : ProfileWarning
+register(username: String, fullname: String, bio: String, type: enumerate("corporate", "ordinary"), image: String) : : String
+login(username: String, password: String) : : bool
Inheritance


Uses >
Uses >
Inheritance
Uses >
Uses >
Uses >
MessageWarning
--
-String type = "message_warning"
-Int message_id [FK to Message]
--
--
+__init__(id:int, reported_by_id: int, user_id: int, reason:String, disputable:int, fine:float, message_id:int)
+get_message() : : Message
+get_warnings_by_messag(message:Message\ ) : : List<MessageWarning>
1
1
1
jobApplication
-Int id
-Int user_id [FK to User]
-Int message_id [FK to Message]
-Int answered
-Int accepted
+__init__(id:int, user_id:int, message_id:int)
+save() : : void
+delete() : : void
+accept() : : void
+deny() : : void
+get_user() : : User
+get_message() : : Message
+get_job_application(id:int\ ) : : jobApplication
+get_job_applications(\ ) : : List<jobApplication>
+get_job_application_obj(row: sqlite3.Row\ ) : : jobApplication
+get_job_applications_by_user(user:User\ ) : : List<jobApplication>
+get_job_applications_by_message(message:Message\ ) : : List<jobApplication>


Tip
-int id
-int user_id [FK to User]
-int author_id [FK to User]
-float amount
+__init__(id:int, user_id:int, author_id:int)
+save() : : void
+delete() : : void
+get_user() : : User
+get_author() : : User
+get_tip(id:int)\ : : Tip
+get_tips()\ : : List<Tip>
+get_tip_obj(row: sqlite3.Row)\ : : Tip
+get_tips_by_user(user:User)\ : : List<Tip>
+get_tips_by_author(author:User)\ : : List<Tip>
Payment
-int id
-int user_id [FK to User]
-float payment_amount
-string explanation
-datetime date
+__init__(id: int, user_id: int, payment_amount: float, explanation: string)
+save() : : void
+get_user() : : User
+validate_card_information(card_number: string, expiration_month: int, expiration_year: int, cvv: int) : : bool
+get_payment(id:int)\ : : Payment
+get_payments()\ : : List<Payment>
+get_payment_obj(row:sqlite3.Row)\ : : Payment
+get_payments_by_user(user:User)\ : : List<Payment>
1
1
1
1
1..n
1..n
1..n
Message
-int id
-int author_id [FK to User]
-String message
-String keywords
-String date
-String time
-String attachment
-float price
-int reads
-int reported
-int original_id [FK to Message]
-int ad = 0
-int job_ad = 0
+__init__(id:int, author_id:int, message:String, keywords:String, attachment:String)
+save() : : void
+delete() : : void
+compute_price() : : float
+get_author() : : User
+get_likes()\ : : List<Like>
+get_dislikes()\ : : List<Dislike>
+get_comments()\ : : List<Comment>
+get_original_message() : : Message
+get_attachment() : : File
+is_trending() : : boolean
+mark_forgiven() : : void
+mark_reported() : : void
+add_read() : : void
+get_message(id:int)\ : : Message
+get_messages()\ : : List<Message>
+get_message_obj(row: sqlite3.Row)\ : : Message
+get_messages_by_author(author:User)\ : : List<Message>
+search_messages(author: String ,keywords: String, likes:tuple, dislikes:tuple)\ : : List<Message>
+get_taboo_words()\ : : List<String>
1


Inheritance
Inheritance
MessageAd
--
int ad = 1
--
+compute_price() : : float <>
+get_ads() : : List<MessageAd>
+get_ads_by_author(author:User\) : : List<MessageAd>
MessagejobAd
--
int ad = 1
int job_ad = 1
--
+compute_price() : : float <>
+get_job_ads() : : List<MessagejobAd>
+get_job_ads_by_author(author:User\) : : List<MessagejobAd>


OrdinaryUser
-int id
-int auth = 1
-String username
-String fullname
-String bio
-String image
-String password
-String salt
-int reviewed
-int denied
-String reason
-float balance
-String profile_picture
-int first_password_changed
-int first_payment_done
-int reported
-int demoted
-int super = 0
-int corporate = 0
--
+__init__(id:int, username:String, fullname:String, bio:String, image:String)
+get_news_feed() : : List<Message>$ <>
+update_password(password:String) : : void
+update_profile_picture(image:String) : : void
+update_information(fullname:String, bio:String) : : void
+deposit_money(amount:String, explanation:String) : : void
+charge_money(amount:String, explanation:String, force_charge:bool) : : void
+post_message(message:String, keywords:String, attachment:String) : : Message
+like_message(message:Message) : : Like
+unlike_message(message:Message) : : void
+dislike_message(message:Message) : : Dislike
+undislike_message(message:Message) : : void
+follow_user(user:User) : : void
+unfollow_user(user:User) : : void
+tip_user(user:User, amount:float) : : Tip
+dispute_warning(warning:Warning) : : void
+apply_to_job(message:MessagejobAd) : : JobApplication
+repost(message:Message) : : Message
+get_warnings() : : List<Warning>
+get_messages() : : List<Message>
+get_followers() : : List<OrdinaryUser>
+get_following() : : List<OrdinaryUser>
+get_likes() : : List<Like>
+get_dislikes() : : List<Dislike>
+get_tips() : : List<Tip>
+get_job_applications() : : List<jobApplication>
+get_payments() : : List<Payment>
+get_comments() : : List<Comment>
+save() : : void
+get_user(id:int\) : : OrdinaryUser
+get_user_by_username(username:String\) : : OrdinaryUser
+get_users() : : List<OrdinaryUser>
+get_user_obj(row: sqlite3.Row\) : : OrdinaryUser
+get_users_by_author(author:OrdinaryUser\) : : List<OrdinaryUser>
+get_taboo_words() : : List<String>
Inheritance


SuperUser
--
-int super = 1
--
+add_taboo_word(word:String) : : void
+remove_taboo_word(word:String) : : void
+accept_user(user:OrdinaryUser) : : void
+reject_user(user:OrdinaryUser) : : void
+forgive_warning(warning:Warning) : : void
+reject_dispute(warning:Warning) : : void
Inheritance
CorporateUser
--
-int corporate = 1
-int super = 0
--
+post_ad(message:String, keywords:String, attachment:String) : : MessageAd
+post_job_ad(message:String, keywords:String, attachment:String) : : MessageJobAd
+accept_application(application:JobApplication) : : void
+reject_application(application:JobApplication) : : void
+get_applications() : : List<JobApplication>
+get_ads() : : List<MessageAd>
+get_job_ads() : : List<MessageJobAd>

Detailed design and class diagrams

System description

The system is formed using the following elements

- A sqlite3 database
- A Flask webserver written in Python
- GUI will be web-browser-based
- HTML/Jinja for templating
- Bootstrap 5.0 for consistent layout
- Javascript for actions that do not require page reload, in the web-browser

The database includes the following tables:

- users
- messages
- comments
- likes
- follows
- tips
- payments
- job_applications
- warnings

The server-side code will include the following classes:

- FlaskApp
- Comment
- CommentWarning
- Config
- CorporateUser
- Database
- Dislike
- File
- JobApplication
- Like
- Message
- MessageAd
- MessageJobAd
- MessageWarning
- OrdinaryUser
- Payment
- ProfileWarning
- SuperUser
- Tip
- User
- Warning

Comment

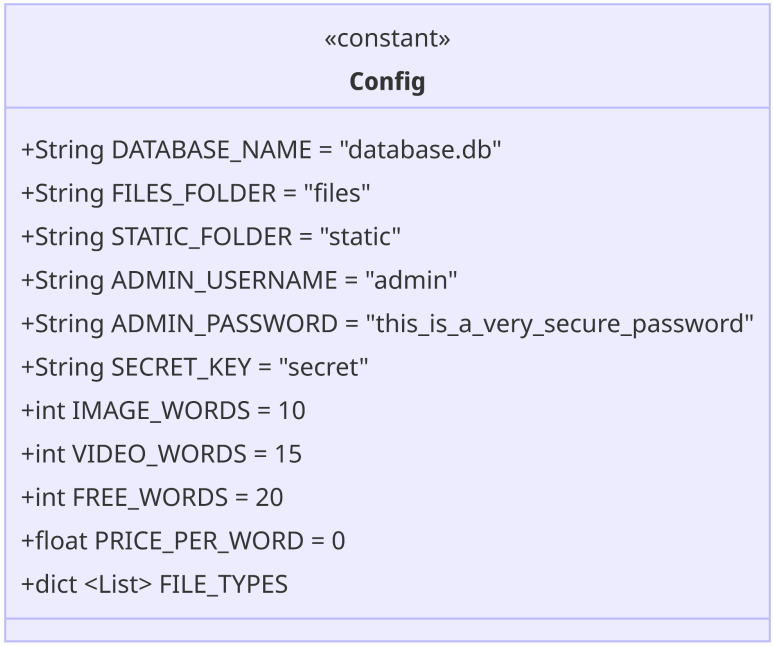
Diagram:

Methods:
__init__(id: int, user_id: int, message_id: int, comment: String)
if id, get comment by id, otherwise save a new comment
save()
save info to the database
delete()
delete comment from the database
get_author()

get the user who commented
get_message()
get the message that was commented
mark_reported()
mark the comment as reported and save
mark_forgiven()
mark the comment as not reported and save
get_comment(id:int)
get comment by id
get_comments()
get all comments
get_comment_obj(row: sqlite3.Row)
create a Comment object from a sqlite3.Row
get_comments_by_user(user:User)
get all comments by a user
get_comments_by_message(message:Message)
get all comments for a message

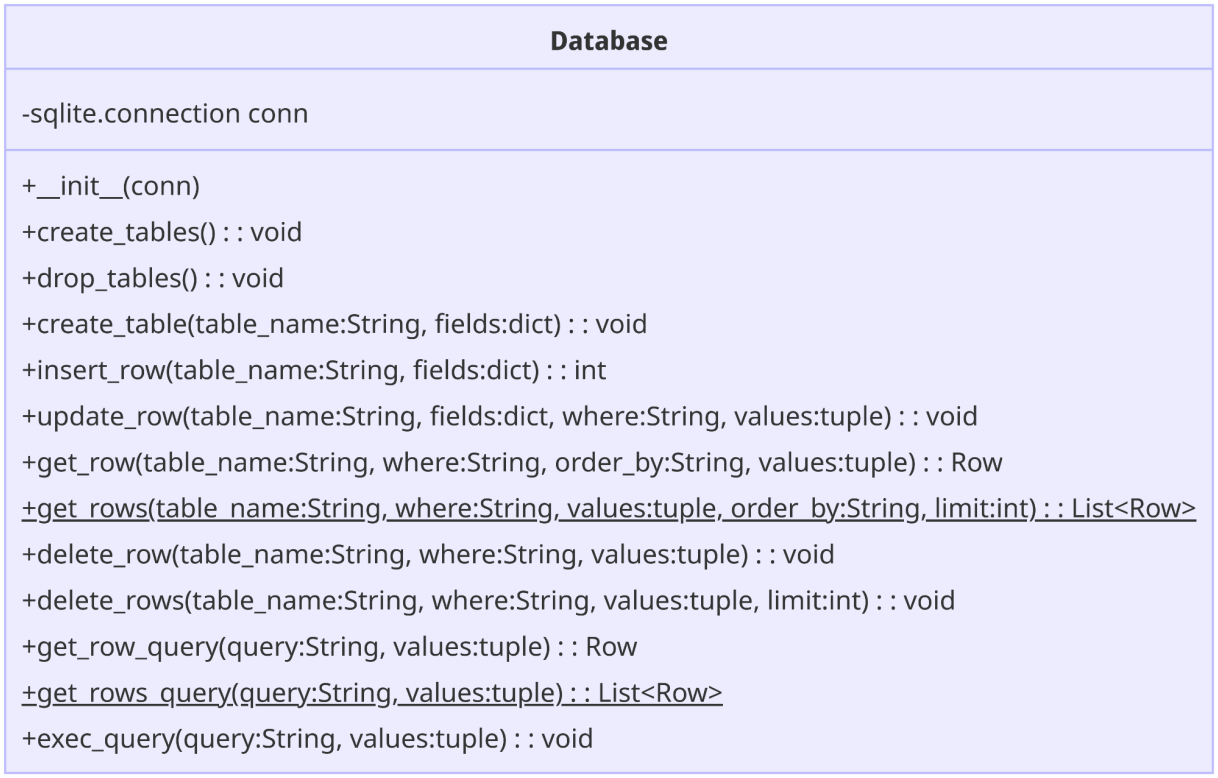
Config

This is only a Config.py file with various settings of the app



Database

Diagram:

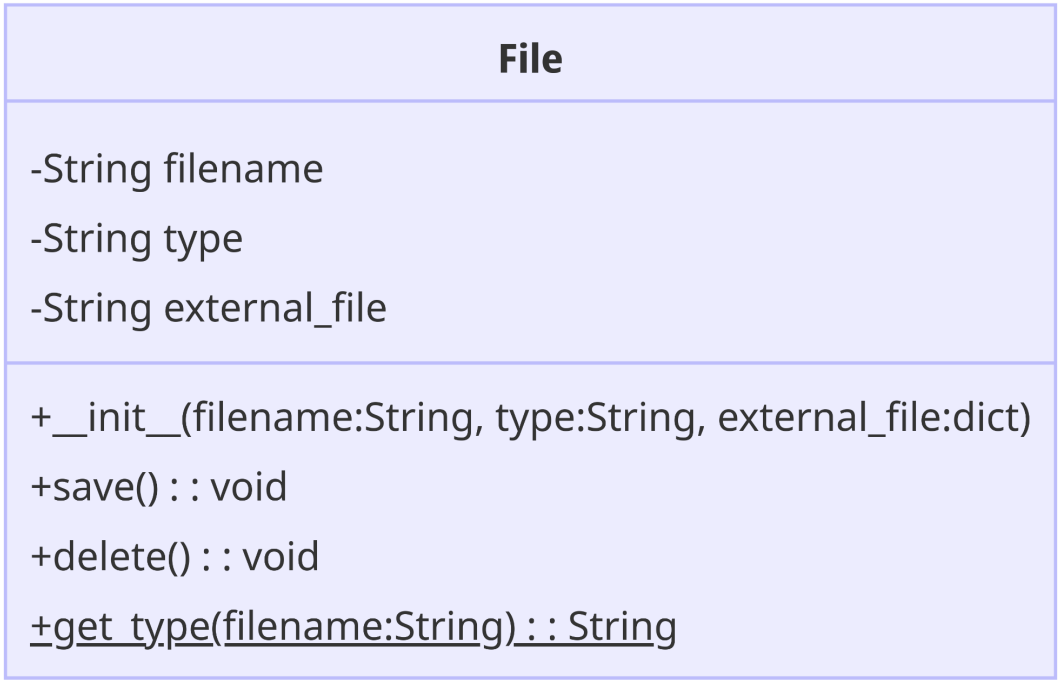


Methods:
__init__(conn)
sets the connection variable
create_tables()
creates the tables in the database
drop_tables()
drop all the tables from the database
create_table(table_name:String, fields:dict)
create a table in the database
insert_row(table_name:String, fields:dict)
insert a row into the database and return its id
update_row(table_name:String, fields:dict, where:String, values:tuple)
update a row in the database
get_row(table_name:String, where:String, order_by:String, values:tuple)
use get_row_query to get a row from the database given the params
get_rows(table_name:String, where:String, values:tuple, order_by:String, limit:int)
use get_rows_query to get rows from the database given the params

delete_row(table_name:String, where:String, values:tuple)
delete a row from the database
delete_rows(table_name:String, where:String, values:tuple, limit:int)
delete rows from the database
get_row_query(query:String, values:tuple)
exec a select and return only one row
get_rows_query(query:String, values:tuple)
exec a select and return all rows
exec_query(query:String, values:tuple)
exec a query

File

Diagram:



Methods:
__init__(filename:String, type:String, external_file:dict)
if filename, get file by filename, otherwise save a new file using external info
if the file is external, validate it's type
save()
if the file is external, save it to the system
delete()
delete file from the system
get_type(filename:String)
get the type of the file by filename

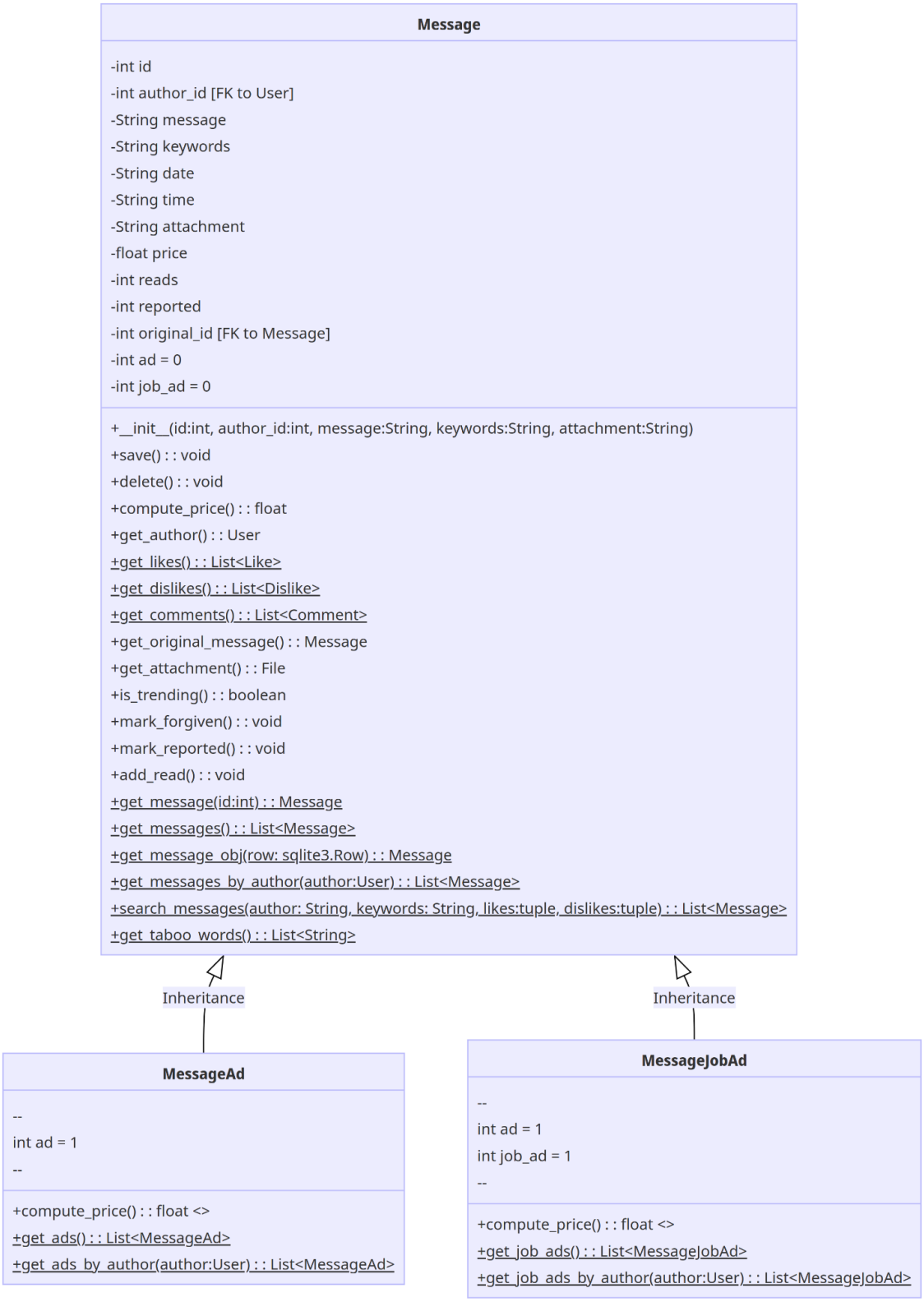
Like, Dislike(Like)

Diagram:

Methods for Like class:
__init__(id:int, user_id: int, message_id:int)
if id, get like by id, otherwise save a new like
save()
save info to the database
delete()
delete like from the database
get_user()
get the user who liked the message
get_message()
get the message that was liked
get_like(id:int)
get like by id
get_likes()
get all likes
get_like_obj(row: sqlite3.Row)
create a Like object from a sqlite3.Row
get_likes_by_message(message:Message)
get all likes for a message
get_likes_by_user(user:User)
get all likes by a user

Additional/Overwriiten methods for Dislike class, which extends Like as they share attributes:
get_dislike(id:int)
get dislike by id
get_dislikes()
get all dislikes
get_dislike_obj(row: sqlite3.Row)
create a Dislike object from a sqlite3.Row
get_dislikes_by_message(message:Message)
get all dislikes for a message
get_dislikes_by_user(user:User)
get all dislikes by a user

Message, MessageAd(Message), MessageJobAd(Message)



Methods for the Message class:

__init__(id:int, author_id:int, message:String, keywords:String, attachment:String)
if id, get message by id, otherwise save a new message
if it is a new message, check and replace the number of taboo words
if it is a new message, check if author has the right to post it (corporate or ordinary)

save()
save info to the database

delete()
delete message from the database

compute_price()
compute the price of the message
take the author status into account (corporate or ordinary)

get_author()
get the user who posted the message

get_likes()
get all likes for the message

get_dislikes()
get all dislikes for the message

get_comments()
get all comments for the message

get_original_message()
get the original message if this is a repost

get_attachment()
get the attachment file

is_trending()
check if the message is trending

mark_forgiven()
mark the message as not reported and save

mark_reported()
mark the message as reported and save

add_read()
add a read to the message

get_message(id:int)
get message by id

get_messages()
get all messages

get_message_obj(row: sqlite3.Row)
create a Message object from a sqlite3.Row

get_messages_by_author(author:User)
get all messages by a user

search_messages(author: String, keywords: String, likes:tuple, dislikes:tuple)
search messages by author, keywords, number of likes and dislikes (min,max)

get_taboo_words()
get all taboo words

MessageAd and Message Job Ad extend message. The main difference between them is that MessageAd has the internal ad variable set to true and users can apply to that “message”.
Additional/Overwritten Methods for these classes:
compute_price()
compute the price of the message, take into account that it is an ad
get_ads()
get all ads
get_ads_by_author(author:User)
get all ads by a user

JobApplication

Diagram:

Methods:
__init__(id:int, user_id:int, message_id:int)
if id get application by id, otherwise save a new application
save()
save info to the database
delete()
delete application from the database
accept()
accept the application
deny()
deny the application
get_user()
get the user who applied
get_message()
get the message to which the user applied
get_job_application(id:int)
get application by id
get_job_applications()
get all applications
get_job_application_obj(row: sqlite3.Row)
create a JobApplication object from a sqlite3.Row
get_job_applications_by_user(user:User)
get all applications by a user
get_job_applications_by_message(message:Messaage)
get all applications for a message

User, OrdinaryUser(User), CorporateUser(OrdinaryUser), SuperUser(CorporateUser)

Methods for the User class:

get_news_feed()
get trending posts first, others after
get_trending_messages()
get trending posts
search_messages(author: String, keywords: String, likes:tuple, dislikes:tuple)
search posts by author, keywords, number of likes and dislikes (min,max)
report_message(message)
report a message
issue a warning to the author
check if the author has more than 3 warnings and take action if that's the case
report_ad(message)
report a message as an ad posted by a regular user
issue a warning to the author
fine the author 10
check if the author has more than 3 warnings and take action if that's the case
report_comment(comment: Comment)
report a comment
issue a warning to the author
check if the author has more than 3 warnings and take action if that's the case
report_profile(user)
report a user
issue a warning to the user
check if the author has more than 3 warnings and take action if that's the case
register(username: String, fullname: String, bio: String, type: enumerate("corporate", "ordinary"), image: String)
register as a user in the system
create a temporary password
return the temporary password
login(username: String, password: String)
login to the system
check if the user exists
check if the password is correct
set the user in the session
return true if the login was successful

Additional/Overwritten methods for OrdinaryUser(User) class:
__init__(id:int, username:String, fullname:String, bio:String, image:String)
if id, get user by id, otherwise save a new user
get_news_feed()
overwrite the method from User

get the posts followed first, then trending, then others
update_password(password:String)
update the password
update_profile_picture(image:String)
update the profile picture
update_information(fullname:String, bio:String)
update the information
deposit_money(amount:String, explanation:String)
deposit money into the account
save the payment
charge_money(amount:String, explanation:String, force_charge:bool)
if force_charge, charge the money
otherwise, check if the user has enough money and only then charge
save the payment
post_message(message:String, keywords:String, attachment:String)
post a message
like_message(message:Message)
like a message
unlike_message(message:Message)
unlike a message
dislike_message(message:Message)
dislike a message
undislike_message(message:Message)
undislike a message
follow_user(user:User)
follow a user
unfollow_user(user:User)
unfollow a user
tip_user(user:User, amount:float)
check if the user has enough money
tip the other user
save the tip
dispute_warning(warning:Warning)
dispute a warning
apply_to_job(message:MessageJobAd)
apply to a job
repost(message:Message)
repost a message
get_warnings()
get all warnings

get_messages()
get all messages
get_followers()
get all followers
get_following()
get all following
get_likes()
get all likes
get_dislikes()
get all dislikes
get_tips()
get all tips
get_job_applications()
get all jobs to which the user applied
get_payments()
get all payments
get_comments()
get all comments
save()
save the user to the database
get_user(id:int)
get user by id
get_user_by_username(username:String)
get user by username
get_users()
get all users
get_user_obj(row: sqlite3.Row)
create an OrdinaryUser object from a sqlite3.Row
get_users_by_author(author:OrdinaryUser)
get all users by a user
get_taboo_words()
get all taboo words

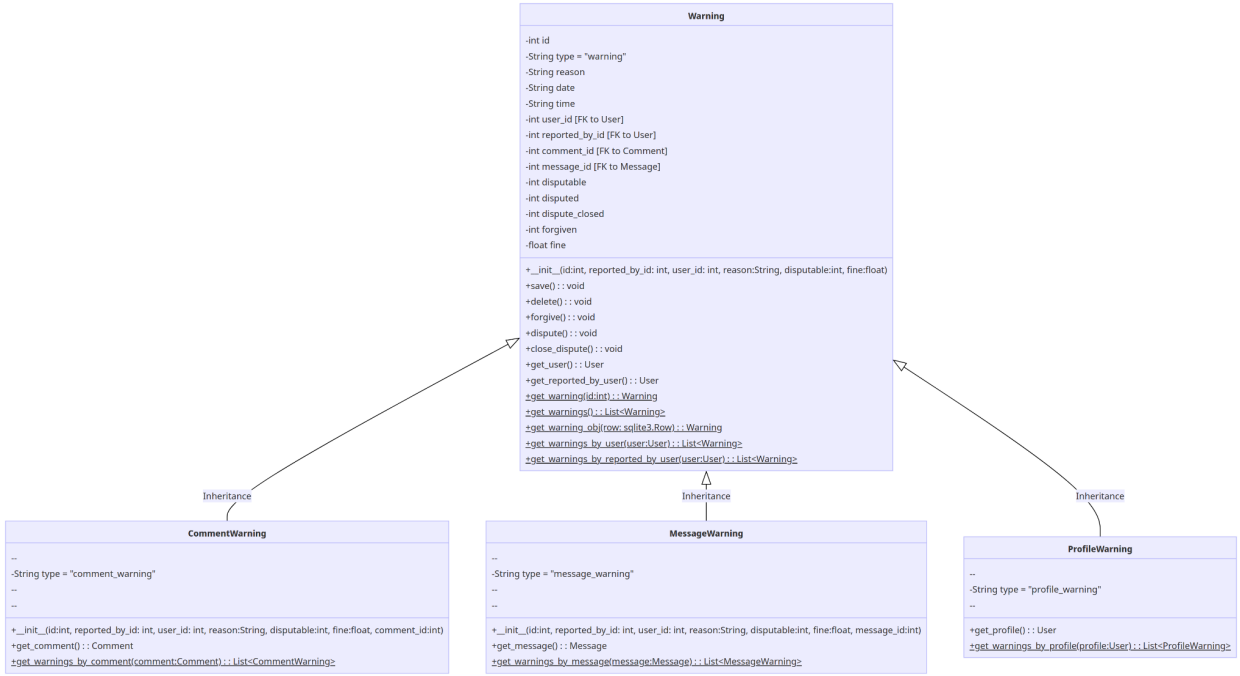
Additional/Overwritten methods for CorporateUser(OrdinaryUser) class:
post_ad(message:String, keywords:String, attachment:String)
post an ad
post_job_ad(message:String, keywords:String, attachment:String)
post a job ad
accept_application(application:JobApplication)
accept a job application

reject_application(application:JobApplication)
reject a job application
get_applications()
get all job applications
get_ads()
get all ads
get_job_ads()
get all job ads

Additional/Overwritten methods for SuperUser(CorporateUser) class:
add_taboo_word(word:String)
add a taboo word
remove_taboo_word(word:String)
remove a taboo word
accept_user(user:OrdinaryUser)
accept a user into the system
reject_user(user:OrdinaryUser)
reject a user from registering into the system
forgive_warning(warning:Warning)
forgive a warning
reject_dispute(warning:Warning)
refuse to forgive a warning

Warning, CommentWarning(Warning), MessageWarning(Warning), ProfileWarning(Warning)

Diagram:



Methods for the Warning class:

__init__(id:int, reported_by_id: int, user_id: int, reason:String, disputable:int, fine:float)

if id, get warning by id, otherwise save a new warning

save()

save info to the database

if the warning is new, fine the user if applicable

delete()

delete warning from the database

forgive()

forgive the warning

refund the fine if applicable

dispute()

dispute the warning

close_dispute()

close the dispute without forgiving the warning

get_user()

get the user who was warned

get_reported_by_user()

get the user who reported the warning

get_warning(id:int)

get warning by id

get_warnings()

get all warnings
get_warning_obj(row: sqlite3.Row)
create a Warning object from a sqlite3.Row
get_warnings_by_user(user:User)
get all warnings for a user
get_warnings_by_reported_by_user(user:User)
get all warnings by a user

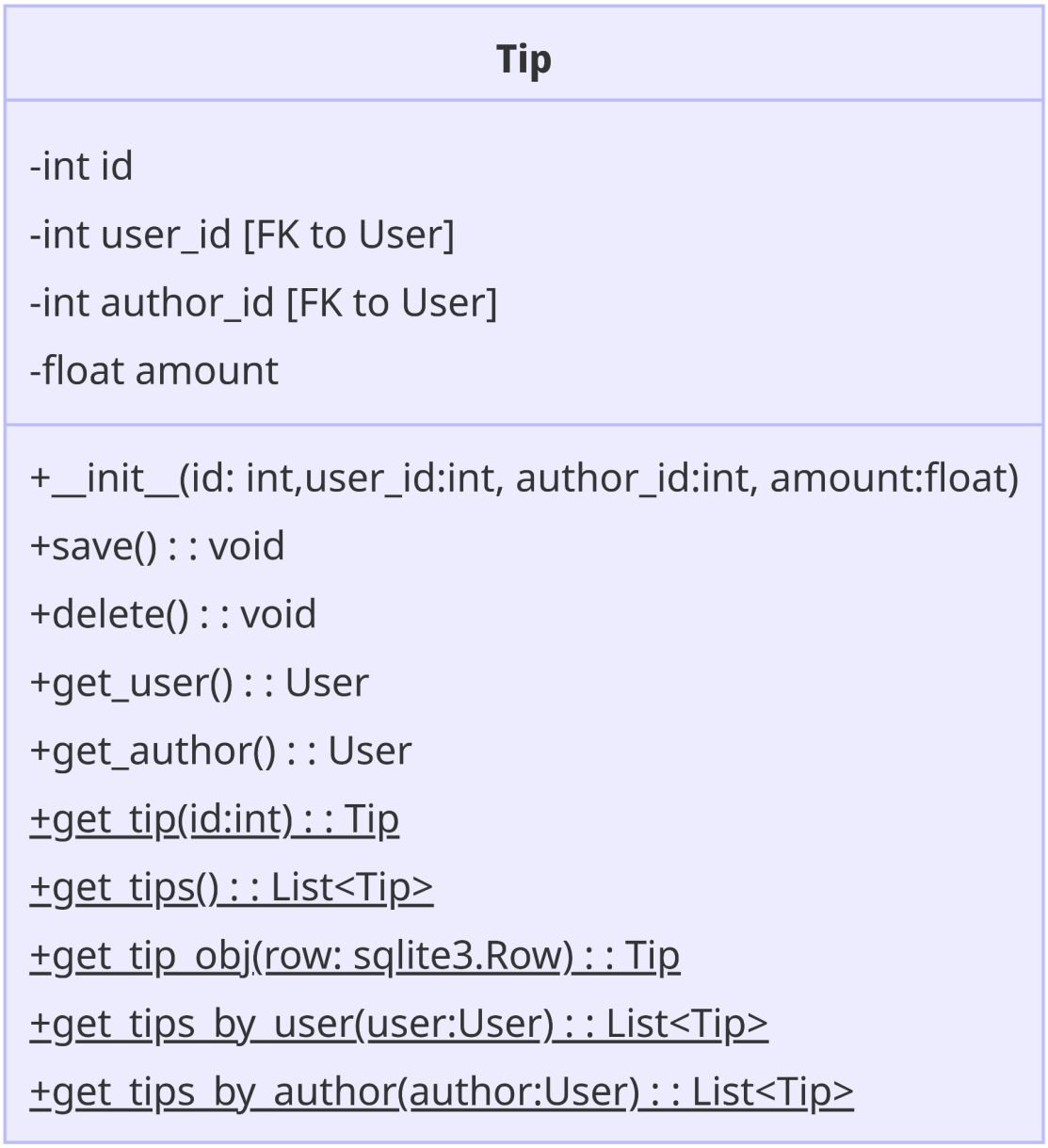
Additional/Overwritten methods for CommentWarning(Warning) class:
get_comment()
get the comment that was warned
get_warnings_by_comment(comment:Comment)
get all warnings for a comment

Additional/Overwritten methods for MessageWarning(Warning) class:
get_message()
get the message that was warned
get_warnings_by_message(message:Message)
get all warnings for a message

Additional/Overwritten methods for ProfileWarning(Warning) class:
get_profile()
alias for get_user from Warning
get_warnings_by_profile(profile:User)
alias for get_warnings_by_user from Warning

Tip

Diagram:

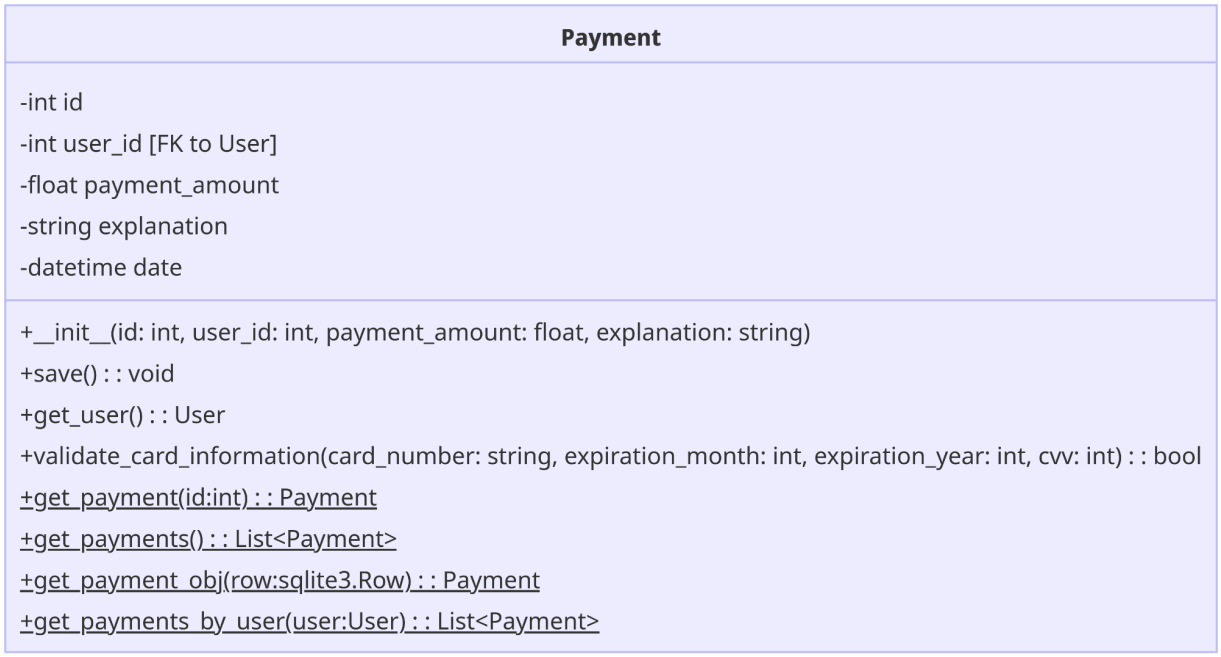


Methods:
__init__(id:int, user_id:int, author_id:int)
if id get tip by id, otherwise save a new tip
save()
save info to the database
delete()
delete tip from the database
get_user()

get the user who tipped
get_author()
get the user who was tipped
get_tip(id:int)
get tip by id
get_tips()
get all tips
get_tip_obj(row: sqlite3.Row)
create a Tip object from a sqlite3.Row
get_tips_by_user(user:User)
get all tips by a user
get_tips_by_author(author:User)
get all tips for a user

Payment

Diagram:



Methods:

__init__(id: int, user_id: int, payment_amount: float, explanation: string)
if id get tip by id, otherwise save a new payment
save()
save info to the database
get_user()
get the user who paid
validate_card_information(card_number: string, expiration_month: int, expiration_year: int, cvv: int)
validate card number to 16 digits
validate expiration month to 1-12
validate expiration year to current year or later
validate cvv to 3 digits
get_payment(id:int)
get payment by id
get_payments()
get all payments
get_payment_obj(row:sqlite3.Row)
create a Payment object from a sqlite3.Row
get_payments_by_user(user:User)
get all payments by a user

FlaskApp


FlaskApp
-app
-g Context
+get_db()
+close_connection(exception)
+static_file(path)
+files(path)
+before_request()
+home_page()
+trending()
+search()
+add_post()
+repost(id)
+post(id)
+like_post(id)
+dislike_post(id)
+comment_post(id)
+remove_comment(id)
+report_post(id)
+report_ad(id)
+report_comment(id)
+delete_post(id)
+applications(id)
+taboo_words()
+add_taboo_word()
+remove_taboo_word()
+users()
+register()
+profile(id)
+login()
+logout()
+update_password()
+report_profile()
+follow_profile()
+unfollow_profile()
+apply_to_job(id)
+accept_user(id)
+reject_user(id)
+accept_application(id)
+reject_application(id)
+add_payment()
+payments()
+add_tip()
+warnings()
+dispute_warning(id)
+forgive_warning(id)
+close_dispute(id)

Methods and routes for the FlaskApp:

get_db()
create a db object if it doesn't exist

close_connection(exception)
close the db connection

static_file(path)
return static files

files(path)
return saved files

before_request()
execute before every request
get the db connection
auth the user if it's in the session
redirect to payments if balance is negative or no deposit has been made

home_page()
show the home page

trending()
show the trending page

search()
search posts by author, keywords, number of likes and dislikes (min,max)

add_post()
check if auth
if GET, show the add_post page
if POST, add the post

repost(id)
repost the post with that id

post(id)
show the post with that id

like_post(id)
check if auth
checked if already liked then unlike and return
like the post with that id

dislike_post(id)
check if auth
checked if already disliked then undislike and return
dislike the post with that id

comment_post(id)
check if auth
get text from post body and post comment

remove_comment(id)
check if auth
delete comment with that id

report_post(id)
report the post with that id

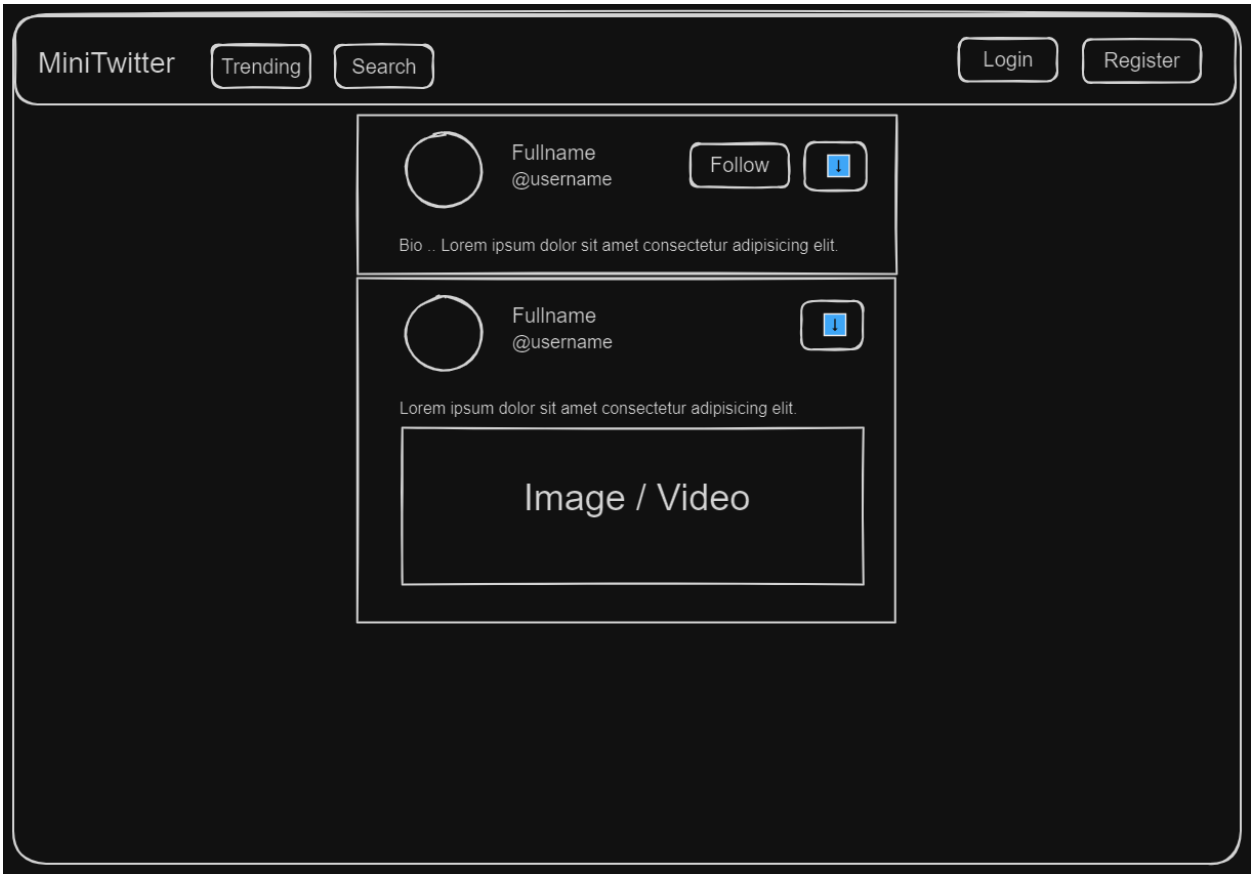
report_ad(id)
report the post with that id as an ad
report_comment(id)
report the comment with that id
delete_post(id)
check if auth and SuperUser or the author of the post
delete post with that id
applications(id)
check if owner of the post
show the applications for that post
taboo_words()
check if SuperUser
show taboo words list
add_taboo_word()
check if SuperUser
add taboo word
remove_taboo_word()
check if SuperUser
remove taboo word
users()
check if SuperUser
show users list
register()
check if auth, redirect to home if already auth
if GET, show the register page
if POST, register the user
profile(id)
show the profile of the user with that id
login()
check if auth, redirect to home if already auth
if GET, show the login page
if POST, try to login
logout()
logout and redirect to home
update_password()
check if auth
if GET, show the update_password page
if POST, update the password
report_profile()
report the profile with that id
follow_profile()
check if auth and not the same user
follow the profile with that id
unfollow_profile()

check if auth and not the same user
unfollow the profile with that id
apply_to_job(id)
check if auth
apply to the job with that id
accept_user(id)
check if SuperUser
accept user with that id
reject_user(id)
check if SuperUser
reject user with that id
accept_application(id)
check if owner of the job ad
accept application with that id
reject_application(id)
check if owner of the job ad
reject application with that id
add_payment()
check if auth
show payment page if GET
deposit money if POST
payments()
check if auth
if SuperUser show all payment history
else show payment history for the user
add_tip(id)
check if auth
tip the user with that id
warnings()
check if auth
if SuperUser show all warning history
else show warning history for the user
dispute_warning(id)
check if auth and warned user
dispute the warning with that id
forgive_warning(id)
check if auth and SuperUser
forgive the warning with that id
close_dispute(id)
check if auth and SuperUser
close the dispute with that id without forgiving the warning

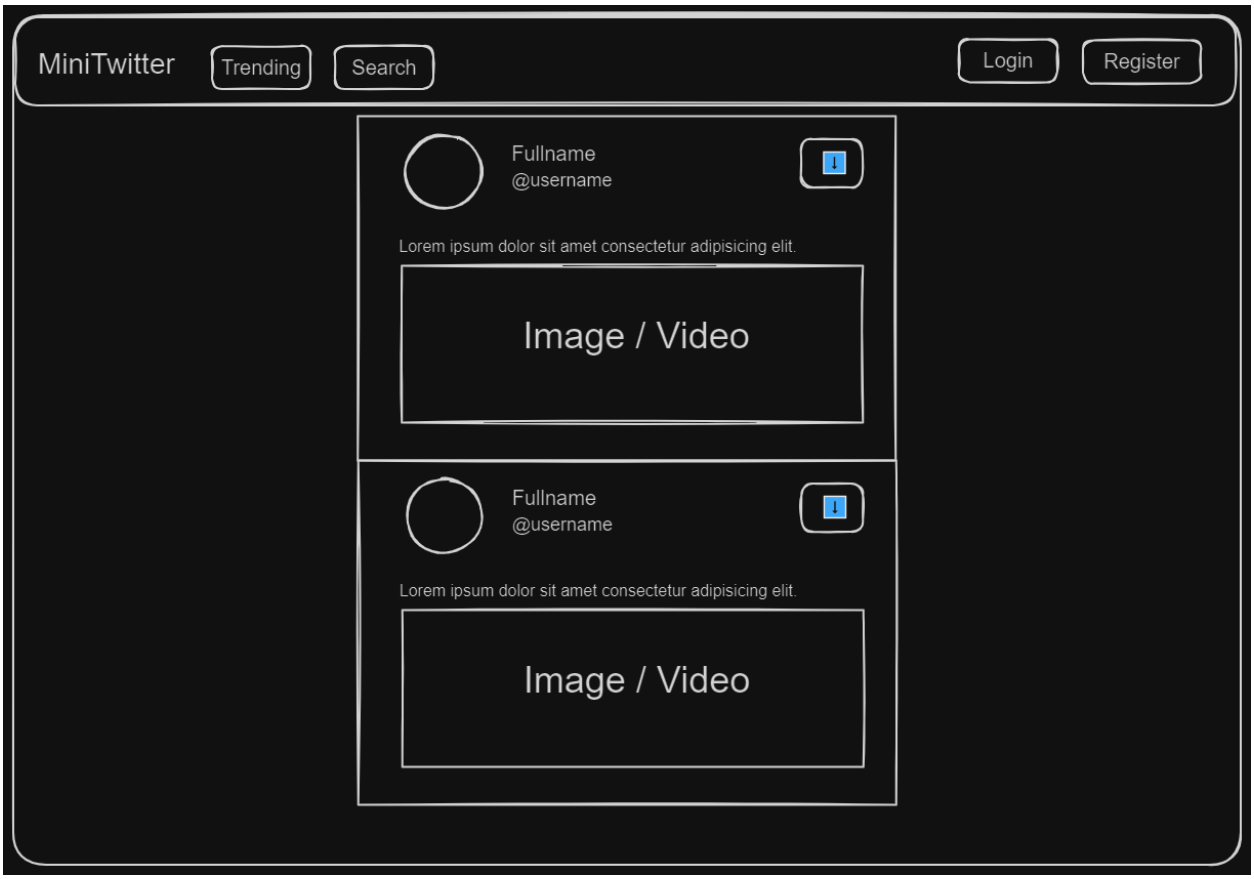
FlaskApp it's not actually a class, it's just named like this for the purpose of this report, in order to be easier to represent it. It's the main python file with the flask server and all of its routes.

System Screen Prototypes

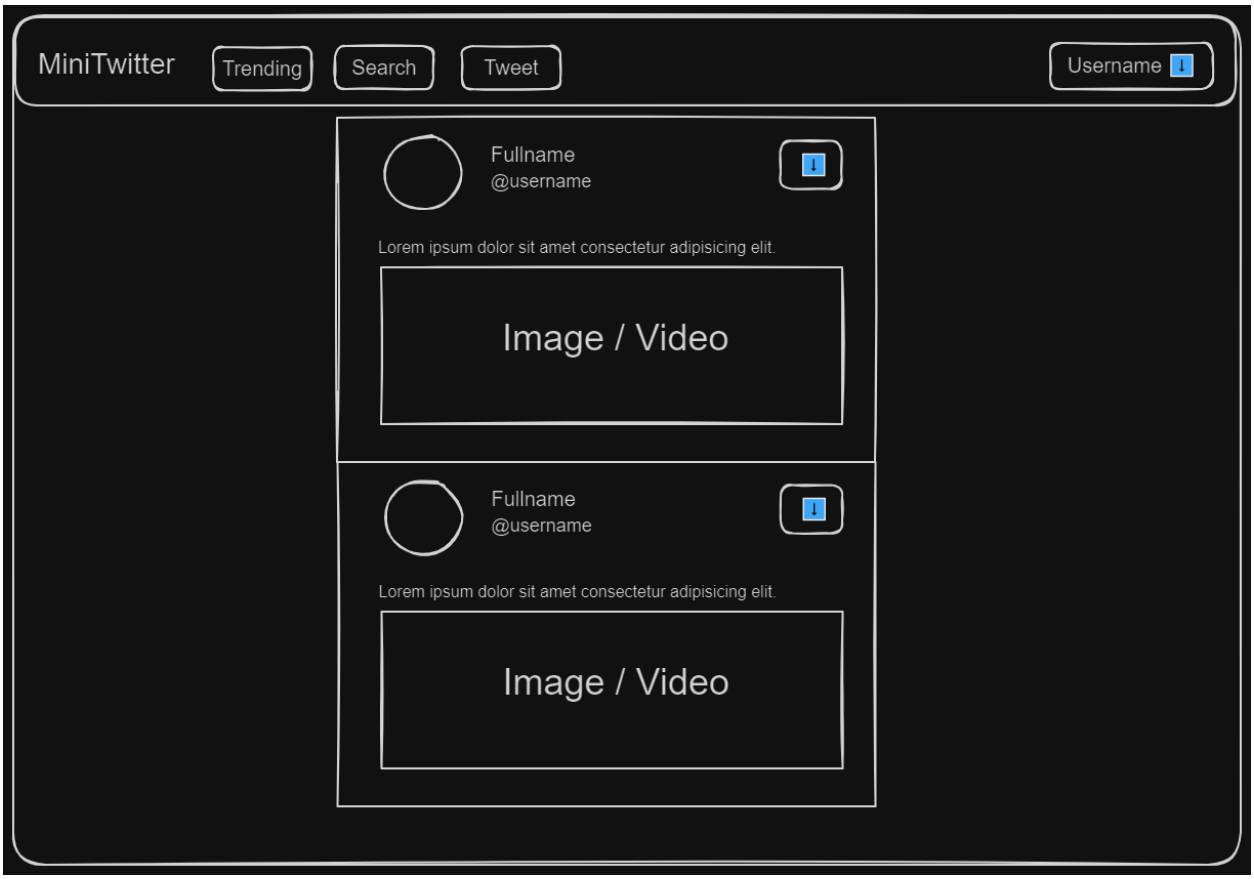
Profile view - Non-authenticated User



News feed view - Non-authenticated User



News feed view - Authenticated User



Group Memos

Date: 11/7/23
Attendees: S.C, F.H, A.Z
Agenda:
-Review progress on the Mini Twitter Design Report.
-Discuss challenges and potential solutions related to system screen prototypes.
-Allocate tasks for the upcoming week.
Discussion Points

S.C Input:
-Highlighted progress on the Flask webserver and sqlite3 database integration.
-Raised concerns about potential bottlenecks in handling simultaneous user actions in the system.

F.H Contribution:
-Shared updates on the GUI development, focusing on browser-based interactions.
-Expressed difficulties with implementing responsive design using Bootstrap 5.0 and requested assistance.

A.Z Observations:
-Provided insights on the E-R Diagram and collaboration class.
-Suggested revisions for the detailed design and class diagrams to enhance system efficiency.

Action Items
S.C:
-To explore optimization techniques for the Flask webserver to manage high user traffic.
-Collaborate with FH on integrating the database with the front-end design.

F.H:
-To continue developing the GUI, incorporating responsive design elements.
-Work with AZ on aligning the front-end design with the system's database structure.

A.Z:
-To refine the E-R Diagram and provide an updated version by the next meeting.
-Assist FH in understanding the database schema for effective GUI integration.
Possible Concerns and Solutions

Communication Gaps:
-Concern: Potential misunderstandings in task allocations and progress updates.
-Solution: Implement a shared digital workspace for real-time updates and clearer communication.

Technical Challenges:
-Concern: Difficulty in synchronizing the database with the user interface.
-Solution: Schedule additional technical workshops to facilitate better understanding and problem-solving.

Time Management:
-Concern: Meeting project deadlines amidst individual workloads.
-Solution: Regular check-ins to assess workload and redistribute tasks if necessary.

Next Meeting
-Date: 11/26/23
-Focus: Review of revised E-R Diagram and progress on GUI integration with the database.

GitHub Link

https://github.com/OarakBabama420/CSC_322_TeamG