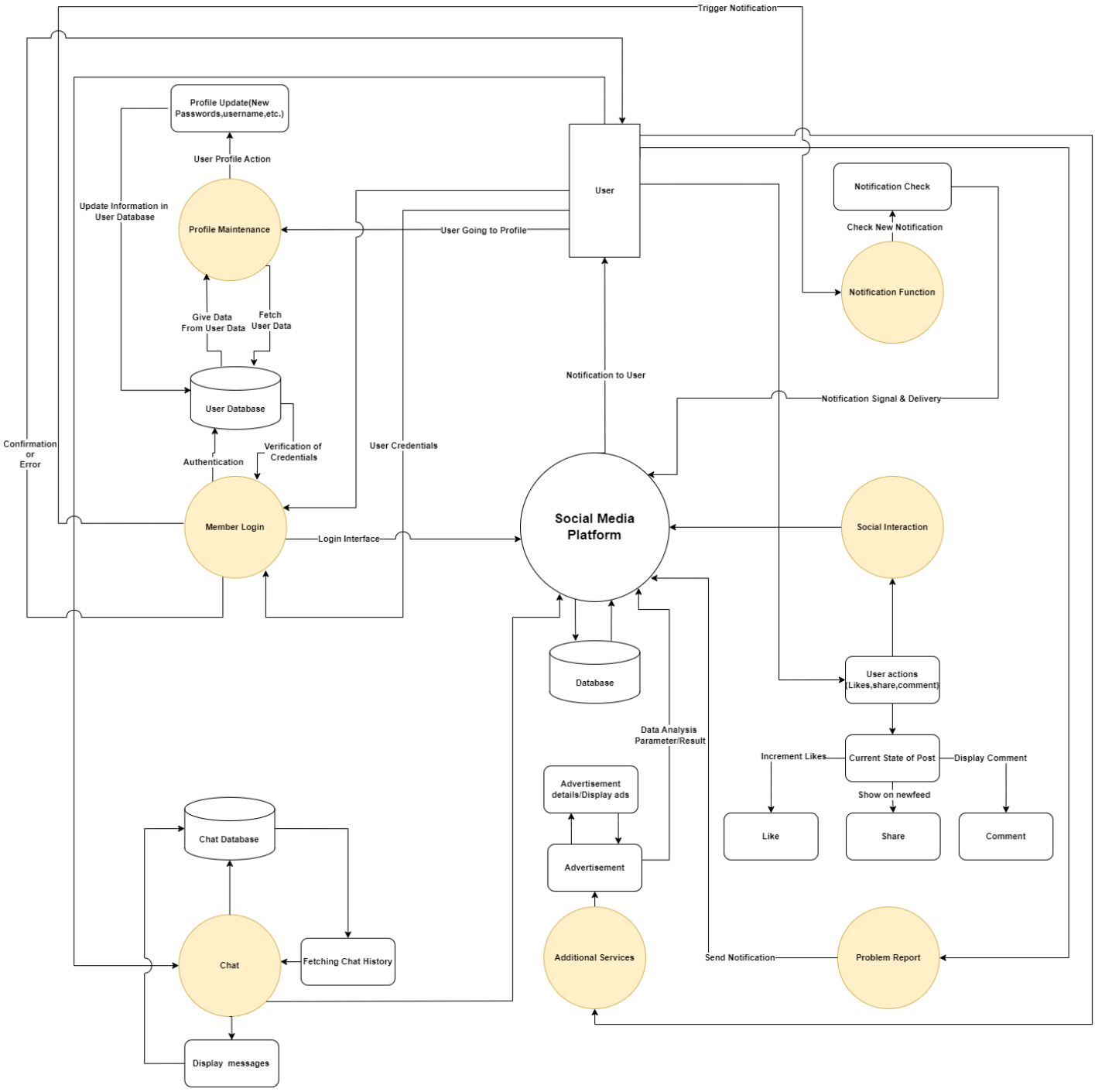
PART 1:

I. DFD:


Trigger Notification
Profile Update(New Passwords,username,etc.)
User Profile Action
Update Information in User Database
Profile Maintenance
User
User Going to Profile
Notification Check
Check New Notification
Notification Function
Give Data From User Data
Fetch User Data
User Database
Notification to User
Notification Signal & Delivery
Confirmation or Error
Authentication
Verification of Credentials
User Credentials
Member Login
Login Interface
Social Media Platform
Social Interaction
Database
User actions (Likes,share,comment)
Data Analysis Parameter/Result
Increment Likes
Current State of Post
Display Comment
Show on newfeed
Advertisement details/Display ads
Like
Share
Comment
Chat Database
Advertisement
Chat
Fetching Chat History
Additional Services
Send Notification
Problem Report
Display messages

a) Member login on social networking platform	**External Input**	User credentials (username and password).
	External Output	Confirmation of successful login or error message.
	External Inquiry	Verification of credentials.
	Internal Logic File	Authentication logic, user session management.
	External Interface FIle	Login interface.
b) Social interaction (like, share, comment)	**External Input**	User actions (like, share, comment).
	External Output	Update to the post (increment likes, display comment, etc.).
	External Inquiry	Fetching the current state of the post.
	Internal Logic File	Logic to process likes, shares, and comments.
	External Interface FIle	Interaction buttons and forms.
c) Messaging/chat functionality	**External Input**	User messages.
	External Output	Display of sent and received messages.
	External Inquiry	Fetching chat history.
	Internal Logic File	Message handling and storage logic.
	External Interface FIle	Chat interface.
d) Personal profile	**External Input**	Profile information updates, new passwords.

maintenance (account/profile information update, password change)	**External Output**	Confirmation of updates or error messages.
	External Inquiry	Fetching current profile information.
	Internal Logic File	Logic to handle profile updates and password changes.
	External Interface FIle	Chat interface.
e) Notification functionality	**External Input**	Triggers for notifications (new message, new comment, etc.).
	External Output	Notifications to users.
	External Inquiry	Checking for new notifications.
	Internal Logic File	Notification logic and delivery mechanism.
	External Interface FIle	Notification display.
f) Additional services (advertisement placement, post data analysis functionality)	**External Input**	Advertisement details, data analysis parameters.
	External Output	Display of ads, analysis results.
	External Inquiry	Fetching relevant data for ads or analysis.
	Internal Logic File	Ad placement logic, data analysis algorithms.
	External Interface FIle	Ad management interface, data analysis tools.
g) Problem reporting	**External Input**	Problem details submitted by users.
	External Output	Confirmation of report receipt, status updates.
	External Inquiry	Fetching status of reported problems.

	Internal Logic File	Problem handling and tracking logic.
	External Interface FIle	Problem reporting form.

II. TCF:

No	FP	(a)	(b)	(c)	(d)	(e)	(f)	(g)
1	**Reliable back-up and recovery**	5	4	4	4	2	1	3
2	**Distributed functions**	4	3	3	3	1	2	1
3	**Heavily used configuration**	4	4	4	1	4	3	2
4	**Operational ease**	4	4	4	4	3	3	3
5	**Complex interface**	3	3	3	3	2	3	2
6	**Reusability**	3	1	1	4	3	4	4
7	**Multiple sites**	4	4	4	4	1	1	1
8	**Data communications**	3	3	3	4	4	3	2
9	**Performance**	4	4	4	3	3	3	2
10	**Online data entry**	4	3	4	4	3	3	3
11	**Online update**	5	2	4	4	4	4	2
12	**Complex processing**	3	3	3	3	2	4	2
13	**Installation ease**	1	1	1	1	1	1	1
14	**Facilitate change**	3	3	3	3	2	3	3
15	**Total**	50	42	45	45	35	38	31
16	**TCF Calculation**	1.15	1.07	1.1	1.1	1	1.03	0.96

TCF Calculation: $TCF = 0.65 + 0.01 * \sum_{i=1}^{14} F_i$

III. FP:

No	Name	Simple	Weighting Factor Average	Complex
1	**External Input**	3	4	6
2	**External Output**	4	5	7
3	**External Inquiry**	3	4	6
4	**Internal File**	5	7	10
5	**External File**	7	10	15

No	Name	(a)	(b)	(c)	(d)	(e)	(f)	(g)
1	**External Input**	3	4	6	4	6	6	3
2	**External Output**	4	5	7	5	7	7	4
3	**External Inquiry**	6	4	6	4	6	6	3
4	**Internal File**	7	7	10	7	10	10	5
5	**External File**	7	10	15	10	15	15	7
6	**Calculations**	27	30	44	30	44	44	27

UFC Calculation: $UFC = \sum_{i=1}^{14} (Number\ of\ items\ of\ variety\ i\ *\ weight_i)$

For UFC calculations we used the python code snippet shown below:

```
    'External Output': [4, 5, 7],
    'External Inquiry': [3, 4, 6],
    'Internal File': [5, 7, 10],
    'External File': [7, 10, 15]
}

component_data = {
    'External Input': [3, 4, 6, 4, 6, 6, 3],
    'External Output': [4, 5, 7, 5, 7, 7, 4],
    'External Inquiry': [6, 4, 6, 4, 6, 6, 3],
    'Internal File': [7, 7, 10, 7, 10, 10, 5],
    'External File': [7, 10, 15, 10, 15, 15, 7]
}
# Using the user-provided complexity assignments (assumed from previous context) and the new weights

complexity_assignments = {
    'a': ['Simple', 'Simple', 'Complex', 'Average', 'Simple'],  # For component (a) and so on
    'b': ['Average', 'Average', 'Average', 'Average', 'Average'],
    'c': ['Complex', 'Complex', 'Complex', 'Complex', 'Complex'],
    'd': ['Average', 'Average', 'Average', 'Average', 'Average'],
    'e': ['Complex', 'Complex', 'Complex', 'Complex', 'Complex'],
    'f': ['Complex', 'Complex', 'Complex', 'Complex', 'Complex'],
    'g': ['Simple', 'Simple', 'Simple', 'Simple', 'Simple']
}

# Complexity indices for easy mapping: "Simple" => 0, "Average" => 1, "Complex" => 2
complexity_indices = {'Simple': 0, 'Average': 1, 'Complex': 2}

# Calculating UFC using new weights and specified complexities
ufc_totals_new_weights = {}

for component, complexities in complexity_assignments.items():
    ufc_total = 0
    for item_type, complexity_level in zip(weights_updated, complexities):
        index = complexity_indices[complexity_level]
```

Then, to finally calculate the FP= TCF * UFC.

No	FP	(a)	(b)	(c)	(d)	(e)	(f)	(g)	Avg FP
1	**TCF**	1.15	1.07	1.1	1.1	1	1.03	0.96	
2	**UFC**	27	30	44	30	44	44	27	
3	**FP**	31.05	32.1	48.4	33	44	45.32	25.92	37.11

IV. Calculating the LOC values

To calculate the LOC values from the FP values provided in part III, we use this following formula

$$LOC = FP \times SLOC\ per\ FP$$

Language	SLOC per FP	FP values	LOC (FP x SLOC per FP)
Assembly Language	320	37.11	11,875.2 ≈ 11,876
C	**128**	**37.11**	**4,750.08 ≈ 4,751**
Cobol	105	37.11	3,896.55 ≈ 3,897
Fortran	105	37.11	3,896.55 ≈ 3,897
Pascal	90	37.11	3,339.9 ≈ 3,340
C++	64	37.11	2,375.04 ≈ 2,376
Ada	70	37.11	2,597.7 ≈ 2,598
php	67	37.11	2,486.37 ≈ 2,487
Java	31	37.11	1,150.41 ≈ 1,151
SQL	12	37.11	445.32 ≈ 446
3GLs	30	37.11	1,113.3 ≈ 1,114
4GLs	20	37.11	742.2 ≈ 743

Assume that the development will be in C, Hence the estimated LOC values will be 4,751.

V.

a.

Cost Driver	Situation	Rating	Value
RELY	Handles sensitive user data (personal information, messages, etc.)	High	1.15
DATA	A social media platform such as Instagram needs a relatively large database size	High	1.15
CPLX	Includes standard social media features	Nominal	1.00

	(profiles, posts, comments, etc.)		
TIME	Requires quick response time for user interactions	High	1.11
STOR	No significant storage constraints	Nominal	1.00
VIRT	Virtual machine environment is stable	Low	0.87
TURN	Standard turnaround times are expected	Nominal	1.00
ACAP	Needs highly skilled personnels	Very High	0.71
AEXP	Needs considerable experience in developing a social media platform	High	0.81
PCAP	Needs highly capable programmers	Very High	0.76
VEXP	Standard level of experience required	Nominal	1.00
LEXP	Programmers should be well-versed in basic programming languages	High	0.91
MODP	Utilizes agile methodologies	High	0.85
TOOL	Advanced software tools for development are needed	High	0.78
SCED	Project schedule should be reasonable	Nominal	1.00

$$EAF = \prod(Cost\ Driver\ Values)$$

$$EAF = 1.15 \times 1.15 \times 1.00 \times 1.11 \times 1.00 \times 0.87 \times 1.00 \times 0.71 \times 0.81 \times 0.76 \times 1.00 \times 0.91 \times 0.85 \times 0.78 \times 1.00$$

$$EAF \approx 0.337$$

b. $Effort\ in\ Person - months = a \times KLOC^{b}$

$Duration\ (time\ for\ development) = c \times Effort^{d}$

Using the semidetatched formula for COCOMO, we get:

$Effort\ in\ Person - months = 3.0 \times (4.8)^{1.12} \approx 17.382\ months$

$Duration\ (time\ for\ development) = 2.5 \times 0.337^{0.35} \approx 1.708\ months$

VI.

a. The tool that we used to re-estimate the size of our planned social media platform is UCP (Use Case Points). Gustav Karner created the UCP technique in 1993. It's very helpful in object-oriented settings because it's made to predict the size of a software project based on use cases. The main functions of UCP is by assessing the quantity and complexity of use cases as well as the players that are a part of the system, UCP calculates effort. To improve the estimate, this approach additionally takes environmental and technical elements into account.

Companies that used the UCP as their tools:

- At home
  UCP has been utilized for project estimation by Indian IT businesses such as Infosys and TCS (Tata Consultancy Services).
- Abroad
  UCP is used by businesses such as IBM and Capgemini for software development project estimating.

We can calculate the UCP as follows:

1. Identify Actors and Use Cases
   Sort actors into 3 different categories: simple, average, and complex, according to how they interact with the system. Sort use cases according to their level of complexity as well.

2. Assign Weights
   Based on each type of actor and use case, assign their weights by following:
   - Simple Actor: 1
   - Average Actor: 2
   - Complex Actor: 3
   - Simple Use Case: 5
   - Average Use Case: 10
   - Complex Use Case: 15

3. Calculate UUCP (Unadjusted Use Case Points)
   $UUCP = Total\ Use\ Case\ Points + Total\ Actor\ Points$

4. Adjust for TCF (Technical Complexity) and EF (Environmental Factors)
   For TCF Calculation, determine the 13 technical factors and give each of them a weight from 0 to 5, then use the formula to determine the value of TCF.
   $TCF = 0.6 + (0.01 \times Sum\ of\ Technical\ Factor\ Weights)$

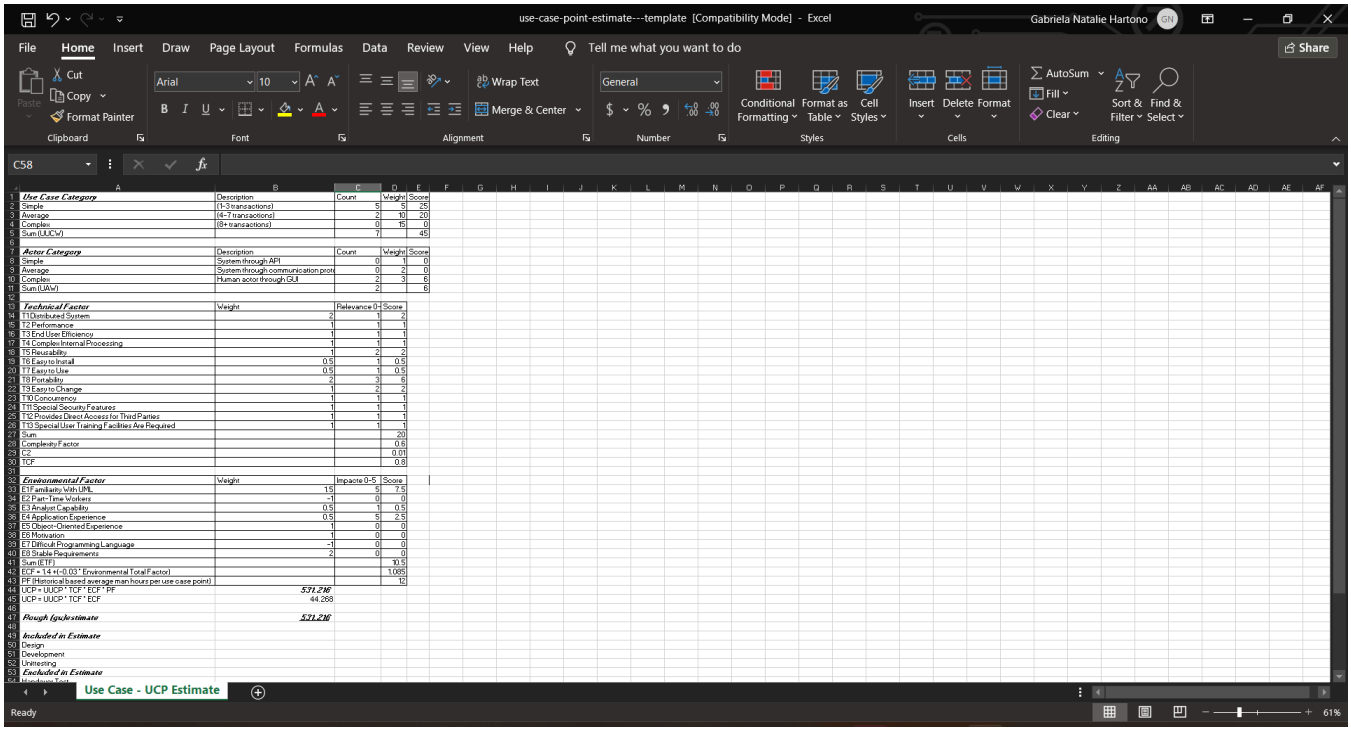
For EF Calculation, similar to TCF Calculation, determine the 8 technical factors and give each of them a weight from 0 to 5, then use the formula to determine the value of EF.

$EF = 1.4 - (0.03 \times Sum\ of\ Environmental\ Factor\ Weights)$

5. Calculate AUCP (Adjusted Use Case Points)

   $UCP = UUCP \times TCF \times EF$

Screenshot of us using it:



b. Similarities between UCP with FP and COCOMO:
   - FP, COCOMO, and UCP are aimed to estimate the size/ effort of a software project.
   - FP, COCOMO, and UCP consider the complexity based on elements. For example, FP considers inputs/outputs and UCP considers cases/actors.

Differences between UCP with FP and COCOMO:
   - Difference in approach is observable as FP focuses on user-facing inputs/outputs and data structures and COCOMO uses a formula-based approach, while UCP focuses on the interactions between cases and actors.
   - FP and UCP provides a measure of size while COCOMO focuses on estimating effort (person-months)
   - FP considers complexity adjustments based on elements, COCOMO considers costs drivers (ex: hardware), and UCP considers environmental factors

c. Advantages of UCP:
    1. Focus on User Requirements: UCP emphasizes user interactions and requirements during the process, making it way more easier for users to engage stakeholders while also ensuring alignment with the business needs.
    2. Straight Forward: UCP is relatively simpler to understand and to use compared to both FP and COCOMO tools, making it accessible to a broader range of projects.
    3. Suitable for Agile Environments: UCP is more adaptable to agile development environments where requirements may evolve rapidly due to its focus on use cases.

    Disadvantages of UCP:
    1. Subjectivity: Although not unique to this specific estimation technique, subjectivity in UCP-based assessments, quality uncertainty of actor points and other variable factors could introduce bias and variance into the final estimates. This may occur if the assessors are unfamiliar with a certain use case, actor, or solution domain.
    2. Limited to use cases: UCP is better fitted for the effort estimation of works with clear use cases. Although it may account for non-functional requirements and other aspects as well. Hence, its applicability is irrelevant for projects that do not comply well with the use case paradigm and works commonly associated with it.
    3. Dependence on Detailed Use Case Documentation: In order for the estimation to be as close to accurate as possible, it is necessary to have detailed and accessible use case documentation, which may not always be available or feasible especially in early project stages.

    Comparison with FP (FP vs. UCP):
    1. Focus: FP measures the challenge on the functional layer of the software system while UCP measures user interactions and requirements
    2. Measurement Units: FP measures size in terms of function points, while UCP uses use case points.
    3. Subjectivity: both metrics require individual assessment, but UCP seems to be aligned with stakeholders due to its measure on the functional level.

    Comparison with COCOMO (COCOMO vs. UCP):
    1. Scope: COCOMO is best suited for the estimation of effort and schedule based on the characteristics of the project to be developed and the actual effort already put into development, setting it aside from UCP, which focuses on one-sided size measurement, namely based on user interactions.

2. Level of detail: COCOMO offers a more detailed approach by incorporating multiple factors influencing project creation, as opposed to UCP, as the former is based on project-dependent parameters, while the latter is largely individual-dependent.
3. Applicability: COCOMO can be applied at all stages of the project, including the conceptual one, due to its broad applicability. In contrast, UCP is suitable only for highly specific projects with limited, yet detailed, functionality regions.

To conclude, UCP should be evaluated with FP and COCOMO in terms of their specialization, measurement characteristics, and audience. UCP focuses on the user interactions and user needs, and FP estimates the complexity of its functions or modules, while COCOMO provides an estimation in person-months, and person-years required for the completion of the project. UCP is easy to understand for the non-professional stakeholders, while FP and COCOMO are designed for software developers and project managers with software engineering knowledge. Finally, the choice of the estimation tool should be made based on the level of project complexity, stakeholder knowledge, and data availability.