Discussion Forum

Fortuna Eyob Fessehaye, fofe8356

1. Sammanfattning

Discussion Forum är ett webbaserat system för att publicera och besvara diskussionsinlägg i en trådad struktur. Systemet är byggt med PHP och MySQL på serversidan samt HTML, CSS och JavaScript på klientsidan. Användare kan skapa inlägg, svara på varandra, och alla svar visas hierarkiskt för att tydligt visa relationen mellan inlägg och kommentarer. Fokus har lagts på enkel interaktion, tydlig layout och strukturerad kod. Projektet demonstrerar hur man kan bygga ett enkelt men kraftfullt forum från grunden.

2. Framtagande

Utvecklingen av **Discussion Forum** har skett stegvis, där varje delmoment byggde vidare på tidigare funktionalitet. Arbetet började med en grundläggande idé om att skapa ett enkelt forum där användare kan publicera inlägg och svara på varandra. Nedan följer fem huvudsakliga steg i utvecklingsprocessen:

2.1 Idé och planering

Projektets mål var att utveckla ett enkelt men funktionellt diskussionsforum där varje inlägg kan få svar och bilda en trådstruktur. Funktionerna skulle vara:

- Starta nya diskussioner genom att fylla i ett formulär.
- Se en lista över tidigare diskussioner.
- Klicka in på en tråd och läsa huvudposten samt dess svar.
- Skriva egna svar eller svara på andra svar (nästlade kommentarer).

För att planera gränssnittet ritades enkla wireframes som visade startsidan med listade inlägg och hur varje inlägg öppnas med svarsfält. I planeringsfasen bestämdes också att en mallstruktur skulle användas för visning av inlägg.

2.2 Databasstruktur

När strukturen för webbapplikationen var planerad skapades två PHP-filer för databasdelen: **create_tables.php** och **db.php**. Den första filen innehåller SQL-kommandon för att skapa två tabeller:

- **posts** - lagrar diskussionsämnen med titel, innehåll och tidsstämpel.
- **comments** - lagrar både direkta svar och nästlade kommentarer. Den innehåller en kolumn parent_id, som möjliggör trådad visning genom att referera till svarets föräldrar.

Denna struktur ger flexibilitet att hantera både huvudinlägg och svar i samma tabell, oavsett svarsnivå. Den andra filen, **db.php**,används för att skapa anslutningen till databasen och inkluderas i alla andra PHP-skript som behöver databaskommunikation.

```php
$sql1 = "CREATE TABLE posts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255) DEFAULT 'Anonymous',
    content TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)";
```

```php
$sql2 = "CREATE TABLE comments (
    id INT AUTO_INCREMENT PRIMARY KEY,
    post_id INT NOT NULL,
    parent_id INT DEFAULT NULL,
    author VARCHAR(255) DEFAULT 'Anonymous',
    comment TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (post_id) REFERENCES posts(id)
)";
```

2.3 HTML-mall

Efter databasstrukturen färdigställdes arbetet med att skapa återanvändbara HTML-mallar för att separera HTML-kod från PHP koden. Förstasidan (index-sidan) består av två huvuddelar: ett formulär för att lägga till nya diskussioner och en tabell som visar alla tidigare diskussioner. I HTML-mallen används platshållare såsom {{discussion_head}} och {{discussion_rows}}. Dessa ersätts dynamiskt i index.php med hjälp av funktionen str_replace(). Det är också i denna fil som formulärdata hanteras och lagras i databasen, samtidigt som sidan genererar tabellinnehållet baserat på aktuella inlägg.
När en användare klickar på ett inlägg skickas denne vidare till view_post.php, där innehållet i det valda inlägget samt dess kommentarer visas. Denna sida bygger på en mallfil, view_post_template.html, med platshållare som {{title}}, {{content}}, {{created_at}} och {{replies}}, vilka fylls i med hjälp av PHP.

```php
// Insert data or remove section
if ($rows === "") {
    $template = str_replace(["{{discussion_head}}", "{{discussion_rows}}"], "", $template);
} else {
    $head = "<th>Discussion</th>\n<th>Started by</th>\n<th>Replies</th>";
    $template = str_replace("{{discussion_head}}", $head, $template);
    $template = str_replace("{{discussion_rows}}", $rows, $template);
}
```

I början skrevs HTML-koden direkt i PHP, men detta ändrades under utvecklingens gång för att istället använda mallfiler och str_replace()-metoden. På så sätt kunde kodens struktur förbättras genom att separera logik från presentationen.

En viktig kod förbättring gällde hanteringen av svar. Inledningsvis hämtades alla kommentarer oavsett struktur, men detta uppdaterades genom att använda fältet parent_id och en rekursiv funktion fetchReplies() för att bygga en trådad struktur. Varje svar genereras då med hjälp av reply_template.html, vilket gör att alla nivåer av svar presenteras konsekvent och tydligt.

2.4 JavaScript för visning av svarsfält

För att förbättra funktionaliteten användes JavaScript för att visa och dölja svarsrutor när användaren klickar på "Reply". För att undvika att olika svarsfält blandas ihop användes olika klasser för huvudkommentarer och svar på kommentarer, till exempel .replyBox och .nested_replyBox.

Vid sidans laddning sätts alla svarsfält som dolda. När användaren klickar på en svarsknapp visas rätt ruta med JavaScript. Det går också att stänga rutan igen genom att klicka på "Cancel".

```
// Hide all reply boxes on load
document.querySelectorAll('.replyBox, .nested_replyBox').forEach(function (box) {
  box.style.display = 'none';
});
```

2.5 CSS för Stil och Struktur

I början fanns bara grundläggande HTML utan någon formgivning, vilket gjorde att allt såg likadant ut och var svårt att läsa. Därför lades CSS till för att ge struktur och förbättra läsbarheten. Varje inlägg fick en tydlig ram (border), och svaren sköts in lite till höger för att visa vilka svar som hör till vilket inlägg. CSS används också för att skapa marginaler och avstånd mellan olika delar, så att innehållet inte krockar visuellt. Designen anpassades för både dator och mobil, så att det fungerar bra oavsett skärmstorlek.

3. Form

I detta projekt består den färdiga applikationen av flera PHP- och HTML-filer som tillsammans möjliggör ett diskussionsforum med stöd för trådade svar. Koden är uppdelad på ett sätt som gör det enkelt att hantera både inlägg, visningar och svarsfunktioner. Nedan beskrivs hur programmet är uppbyggt, med exempel på viktiga kodavsnitt samt pseudokod som visar det övergripande flödet.

3.1 Startsidan – index.html och index.php

Här kan användare skapa ett nytt diskussionsämne och se en lista över alla befintliga diskussioner. Formuläret visas endast när användaren klickar på knappen “Add discussion topic” (hanteras med JavaScript). Själva sidan är uppbyggd som en HTML-mall med platshållare:

Centralt Kodavsnitt:

```php
<?php
header('Content-Type: text/html');
require 'db.php';
// Load the HTML template
$template = file_get_contents("index.html");

// Handle form submission
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['submit_button'])) {
  $subject = trim($_POST['subject']);
  $username = trim($_POST['author']);
  $message = trim($_POST['text_area']);

  if ($subject !== "" && $message !== "" &&  $username!=="") {
    $stmt = $conn->prepare("INSERT INTO posts (title, content, author) VALUES (?, ?, ?)");
    $stmt->bind_param("sss", $subject, $message, $username);
    $stmt->execute();
    $stmt->close();
    // Prevenst resubmission
    header("Location: index.php");
    exit();
  }
}
// Fetch posts from database
$result = $conn->query("SELECT id, title, author, replies FROM posts ORDER BY created_at DESC");
$rows = "";
while ($row = $result->fetch_assoc()) {
    $id = $row['id'];
    $title = htmlspecialchars($row['title']);
    $author= htmlspecialchars($row['author']);
    $replies = $row['replies'];
    $rows .= "<tr><td><a href=\"view_post.php?id=$id\">$title</a></td><td>$author</td><td>$replies</td></tr>\n";
}
// Insert data or remove section
if ($rows === "") {
    $template = str_replace(["{{discussion_head}}", "{{discussion_rows}}"], "", $template);
} else {
    $head = "<th>Discussion</th>\n<th>Started by</th>\n<th>Replies</th>";
    $template = str_replace("{{discussion_head}}", $head, $template);
    $template = str_replace("{{discussion_rows}}", $rows, $template);
}
// Output final result
echo $template;
?>
```

Pseudokod

Set content type to HTML
Include database connection (db.php)
Load the contents of "index.html" into a variable called template

If the HTTP request method is POST and the submit button is pressed then
  Read subject, author and text_area from the form
  If subject, author and message are not empty then
    Insert the values into the posts table (title, content, author)
    Redirect to index.php to prevent form resubmission
    Exit the script

Fetch all rows from the posts table (newest first) and store in a variable “result”
Initialize an empty string variable “rows”

For each row in the result:
    Extract id, title, author and replies
    Append a new table row to rows with:
     - A link to view the post using its ID
     - The author’s name
     - The number of replies

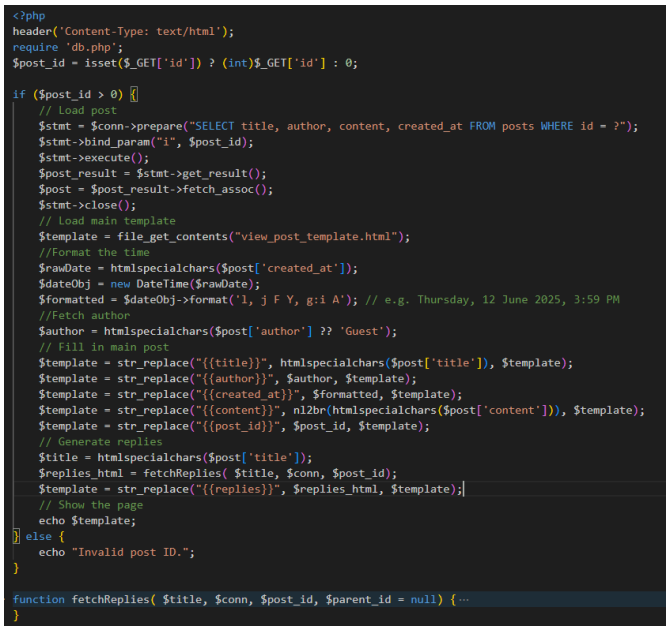
If there are no posts (rows is empty) then
  Remove the placeholders {{discussion_head}} and {{discussion_rows}} from the template
else
  Replace {{discussion_head}} with a row of table headers:
    - "Discussion", "Started by", and “Replies"
  Replace {{discussion_rows}} with the generated HTML rows

Display the final HTML page

3.2 Inläggsvisning – view_post.php och mallarna

När en användare klickar på en tråd i tabellen på startsidan, hamnar hen på view_post.php?id=.... Denna sida ansvarar för att visa själva inlägget samt alla svar. För att strukturera presentationen används en mallfil (view_post_template.html) och varje svar (och svar på svar) byggs upp med hjälp av reply_template.html.

Centralt Kodavsnitt:



```php
<?php
header('Content-Type: text/html');
require 'db.php';
$post_id = isset($_GET['id']) ? (int)$_GET['id'] : 0;

if ($post_id > 0) {
    // Load post
    $stmt = $conn->prepare("SELECT title, author, content, created_at FROM posts WHERE id = ?");
    $stmt->bind_param("i", $post_id);
    $stmt->execute();
    $post_result = $stmt->get_result();
    $post = $post_result->fetch_assoc();
    $stmt->close();
    // Load main template
    $template = file_get_contents("view_post_template.html");
    //Format the time
    $rawDate = htmlspecialchars($post['created_at']);
    $dateObj = new DateTime($rawDate);
    $formatted = $dateObj->format('l, j F Y, g:i A'); // e.g. Thursday, 12 June 2025, 3:59 PM
    //Fetch author
    $author = htmlspecialchars($post['author'] ?? 'Guest');
    // Fill in main post
    $template = str_replace("{{title}}", htmlspecialchars($post['title']), $template);
    $template = str_replace("{{author}}", $author, $template);
    $template = str_replace("{{created_at}}", $formatted, $template);
    $template = str_replace("{{content}}", nl2br(htmlspecialchars($post['content'])), $template);
    $template = str_replace("{{post_id}}", $post_id, $template);
    // Generate replies
    $title = htmlspecialchars($post['title']);
    $replies_html = fetchReplies( $title, $conn, $post_id);
    $template = str_replace("{{replies}}", $replies_html, $template);
    // Show the page
    echo $template;
} else {
    echo "Invalid post ID.";
}

function fetchReplies( $title, $conn, $post_id, $parent_id = null) {···
}
```

Pseudokod

Set content type to HTML
Include the database connection (db.php)

Get the post ID from the URL query string (as an integer)

If the post ID is greater than 0 then
    Prepare and execute a SQL query to fetch the post's title, content, and creation time
    Load view_post_template.html into variable
    Format the creation time into a readable string
    If author is missing then, set it to "Guest"
    Replace placeholders in the template:
        - {{title}} with the post's title
        - {{author}} with the author name
        - {{created_at}} with the formatted time
        - {{content}} with the post content (converted to HTML with line breaks)
        - {{post_id}} with the current post ID
    Call fetchReplies(title, connection, post_id) to build all threaded replies
    Replace {{replies}} in the template with the result from fetchReplies
    Output the final page
else
    Display the message "Invalid post ID"

Function fetchReplies(post_id, parent_id):
    Prepare a SQL query to fetch all comments for the given post_id
    If parent_id is null then

fetch only top-level comments (parent_id IS NULL)
else
fetch replies where parent_id = given parent_id
Execute the query

Load reply_template.html
For each comment in the result:
Replace placeholders in the template
Recursively call fetchReplies(...) to get child replies for this comment
Insert child replies into the {{replies}} placeholder
Append the filled-in reply HTML to a result string
Return the combined HTML string

3.3 Databasen (db.php, create_tables.php)

Här hanteras anslutningen till databasen och skapandet av tabellerna *'posts'* och *'comments'*. Tabellerna skapas via PHP i filen '*create_tables.php*', medan *'db.php'* används för att koppla upp mot databasen och återanvänds i övriga PHP-sidor.

Centralt Kodavsnitt:

```php
<?php
  $host = 'mysql.dsv.su.se';
  $user = 'xxxxxx';
  $pass = 'xxxxxx';
  $dbname = 'xxxxx'; // name of your database


  // Connect to the MySQL
  $conn = new mysqli($host, $user, $pass, $dbname);

  // Check connection
  if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
  }
?>
```

```php
<?php
require 'db.php';

// Drop existing tables if they exist (drop comments first because it references posts)
$conn->query("DROP TABLE IF EXISTS comments");
$conn->query("DROP TABLE IF EXISTS posts");

$sql1 = "CREATE TABLE posts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255) DEFAULT 'Anonymous',
    content TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    replies INT DEFAULT 0
  )";

// Execute first table creation
if ($conn->query($sql1) === TRUE) {
  echo "Table 'posts' created successfully.<br>";
} else {
  echo "Error creating 'posts': " . $conn->error . "<br>";
}

$sql2 = "CREATE TABLE comments (
```

Pseudokod - db.php

Set database credentials: host, username, password, and database name
Connect to MySQL using the provided credentials

If connection fails:
  Stop execution and show error message

Pseudokod - create_post.php

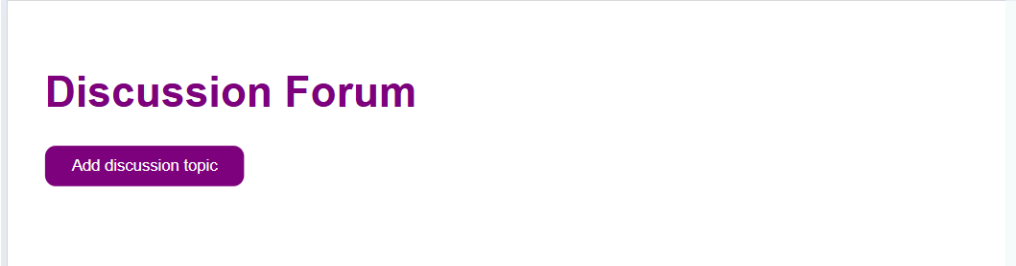
Include db.php (database connection)
Drop "comments" and "posts" tables if they exist
Create "posts" table with id, title, author, content, created_at
If creation is successful then
    Show success message
else
    Show error
Create "posts" table with id, post_id, parent_id, author, comment, created_at
If creation is successful then
    Show success message
else
    Show error

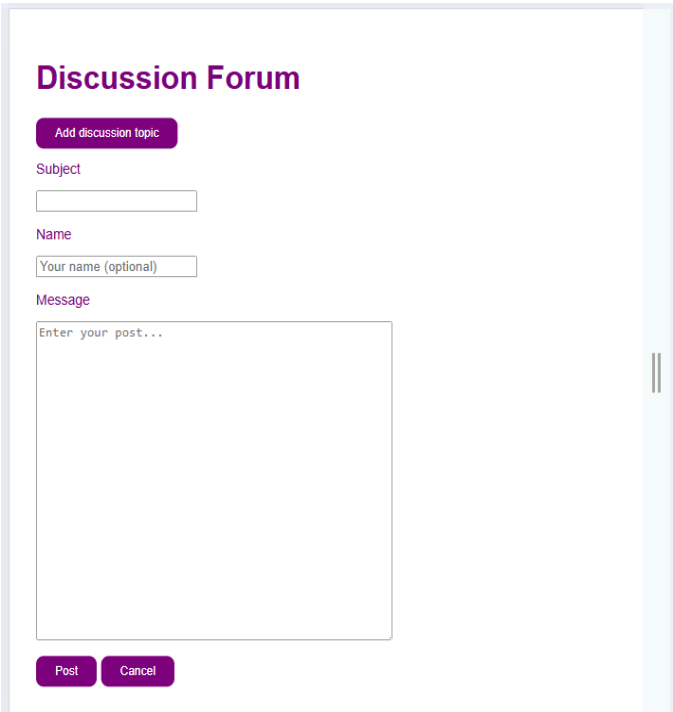
4. Function

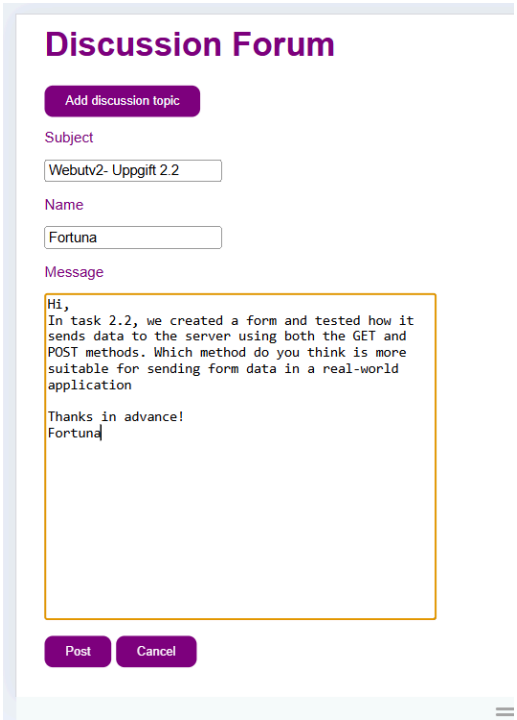
4.1 Startsidan(index.php)

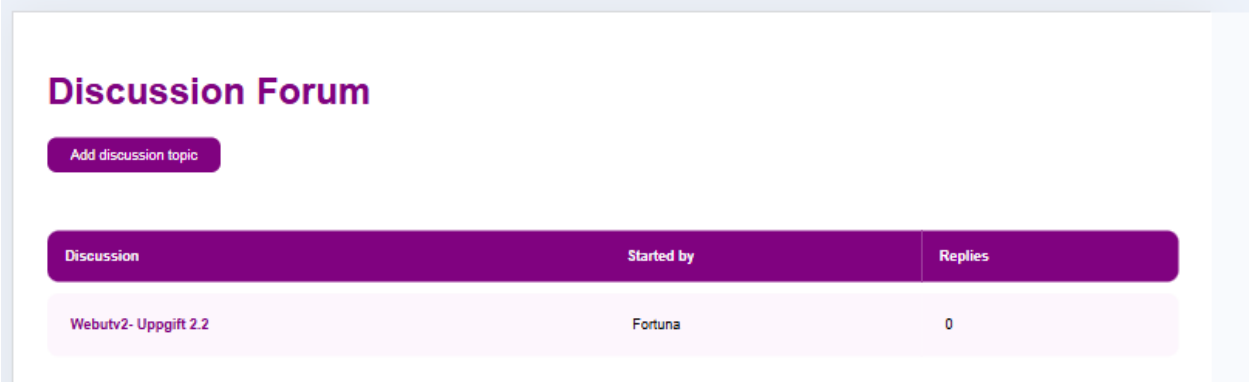
När användaren öppnar webbplatsen visas startsidan (index.php) automatiskt. Sidan innehåller en rubrik, en knapp för att starta en ny diskussion (***Add discussion topic***) samt en tabell med alla tidigare diskussioner. Formuläret för att lägga till en ny tråd är till en början dolt och visas först när användaren klickar på knappen. Detta hanteras med JavaScript som togglar visning av formuläret.

När ett formulär skickas in med titel och innehåll, sparas dessa i databasen i tabellen posts. Användaren omdirigeras därefter till samma sida, där det nya inlägget nu visas högst upp i listan. Alla inlägg visas som en klickbar lista med subject, author och replies.



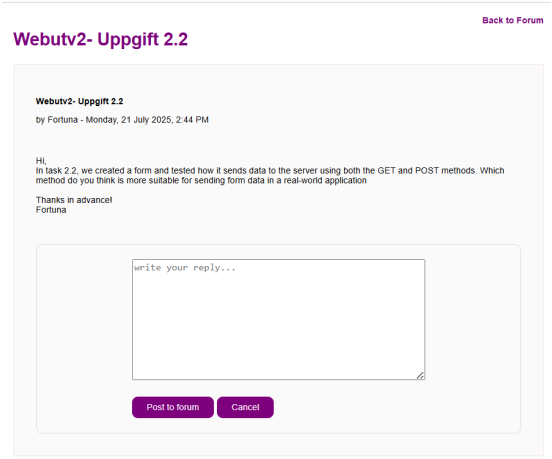


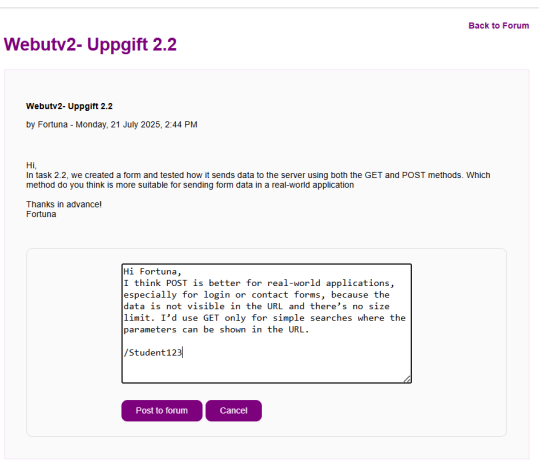




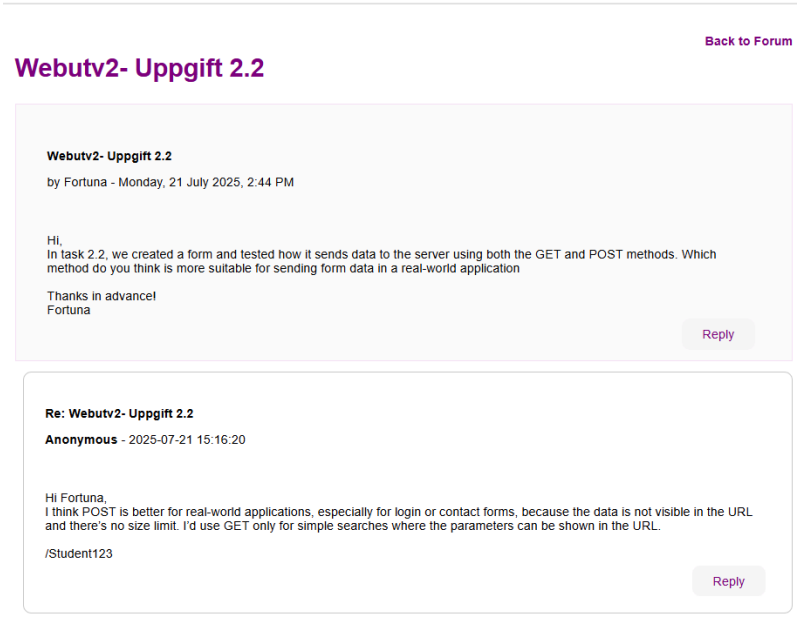
4.2 Trådvisning (view_post.php)

När användaren klickar på en tråd i listan kommer de vidare till *view_post.php?id=....* Där visas hela inlägget, inklusive titel, innehåll och tid. Under inlägget finns ett svarsknapp (*Reply*) som öppnar ett dolt formulär. När detta formulär skickas in sparas svaret i tabellen ‘*comments’*, och användaren omdirigeras tillbaka till samma tråd.

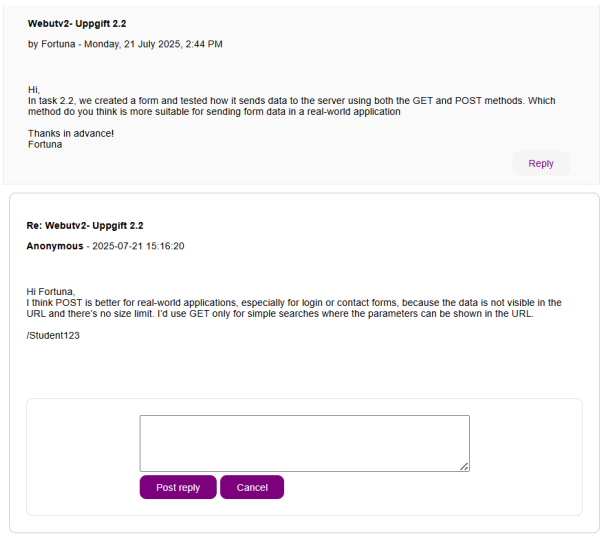


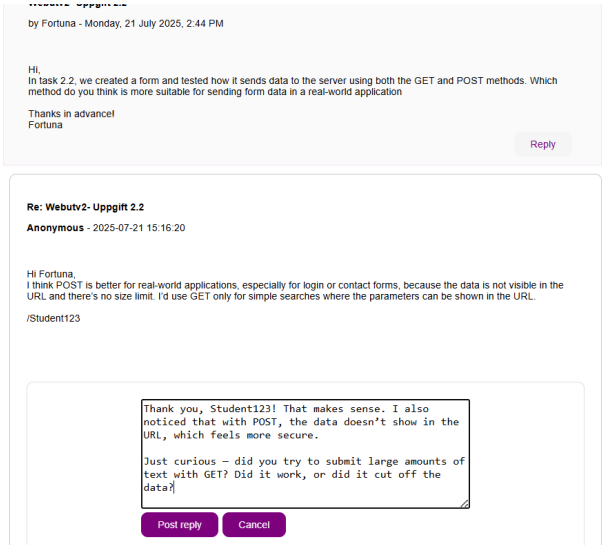


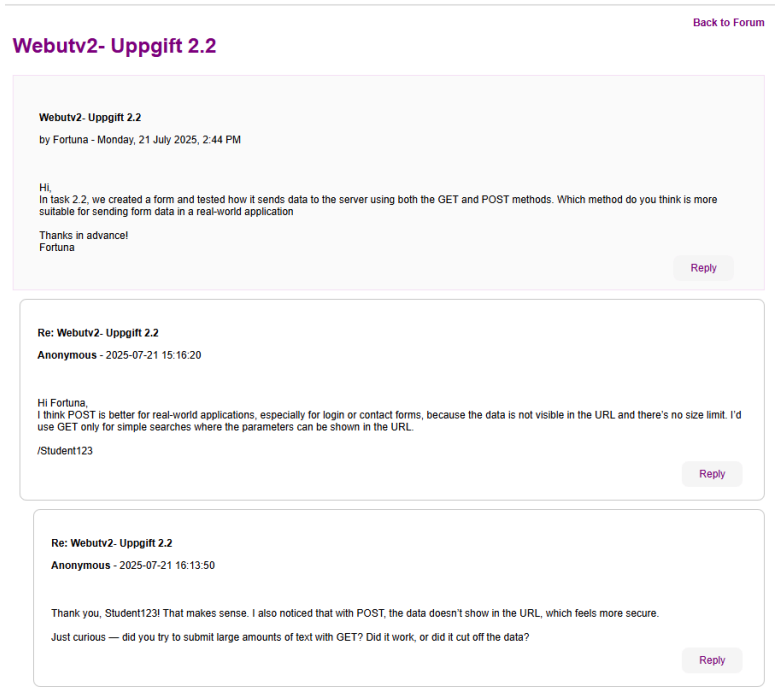
Alla svar som hör till inlägget laddas och visas i ordning. Programmet använder en funktion fetchReplies() som laddar svar rekursivt. Det innebär att ett svar kan ha egna svar, som också visar indraget under sin förälder. För att visa detta visuellt används CSS med indrag, kantlinjer och marginaler.



Varje svar kan i sin tur besvaras på. Under varje kommentar finns därför en "Reply"-knapp som visar ett nytt formulär för just det svaret. Genom att varje formulär skickas med ett parent_id till reply.php, går det att spara svaret som ett barn till rätt kommentar. Vid nästa laddning av sidan läses alla svar in igen, och funktionen fetchReplies() bygger HTML med hjälp av reply_template.html så att strukturen bibehålls. För att undvika rörligt gränssnitt är alla svarsfält från början dolda, och visas först när rätt knapp klickas. Detta görs med Javascript som söker upp rätt .replyBox eller .nested_replyBox och visar den.







När användaren är klar kan knappen 'Back to forum' användas för att gå tillbaka till huvudsidan. Där kan ses uppdaterat antalet svar på varje inlägg.