

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
КЕРУВАННЯ КОМЕНТАРЯМИ ВІДВІДУВАЧІВ МУЗИЧНОГО
ФЕСТИВАЛЮ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
MODERATING THE COMMENTS OF MUSIC FESTIVAL VISITORS

Виконав: студент(ка) 2 курсу, групи КНТ-210м
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

Медведєв Д.В.

(прізвище та ініціали)

Керівник Скрупський С.Ю.

(прізвище та ініціали)

Рецензент Голуб Т.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Медведева Дениса Володимировича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів керування коментарями відвідувачів музичного фестивалю
Research and Software Implementation of Methods for Moderating the Comments of Music Festival Visitors

керівник проєкту (роботи) Скрупський Степан Юрійович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “5” листопада 2021 року № 435

2. Строк подання студентом проєкту (роботи) 13 грудня 2021 року
 3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Скрупський С.Ю., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 7 ” вересня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка архітектури програми	6 тиждень	Розділ 3
4	Розробка програми	7-9 тижні	Розділ 4
5	Тестування та експериментальне дослідження програми	10-11 тижні	Розділ 5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
7	Захист роботи	16 тиждень	

Студент(ка)

(підпис) Медведєв Д.В.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Скрупський С.Ю.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра: 96 с., 33 рис., 2 табл., 3 дод., 21 джерело.

МУЗИЧНИЙ ФЕСТИВАЛЬ, ВІДВІДУВАЧ ФЕСТИВАЛЮ, КОМЕНТАР, ФОРУМ, МАШИННЕ НАВЧАННЯ, КЕРУВАННЯ КОМЕНТАРЯМИ, ОБРОБЛЕННЯ ПРИРОДНОЇ МОВИ.

Об'єкт дослідження – процес керування онлайнвзаємодією відвідувачів музичного фестивалю.

Предмет дослідження – методи керування коментарями відвідувачів музичного фестивалю.

Мета роботи – розробити програмне забезпечення для отримання інформації про музичний фестиваль і комунікації його відвідувачів з засобами керування для убезпечення музичного фестивалю від репутаційних та економічних втрат через дискримінаційні висловлювання відвідувачів.

Матеріали, методи та технічні засоби: мова програмування PHP, мова програмування Python, модель GloVe, метод Random Forest.

Результати. Результатом виконаної роботи є розроблене програмне забезпечення та запропонований метод. Наукова новизна роботи полягає в запропонованому методі керування коментарями відвідувачів музичного фестивалю, який ґрунтується на виконанні оброблення коментаря користувача моделлю GloVe з подальшою класифікацією методом Random Forest, що дозволяє визначати керуюче рішення з метою уникнення репутаційних втрат для фестивалю.

Висновки. Розроблене програмне рішення дозволяє об'єднувати можливості онлайнкомунікації з придбанням квитків на музичний фестиваль для надання відповідних засобів уникнення проблемних відвідувань.

Галузь використання – середні та великі музичні фестивалі.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 96 pages, 33 figures, 2 tables, 3 appendixes, 21 sources.

MUSIC FESTIVAL, FESTIVAL VISITOR, COMMENT, FORUM, MACHINE LEARNING, COMMENT MANAGEMENT, NATURAL LANGUAGE PROCESSING.

The object of research is the process of managing online interaction of music festival visitors.

The subject of research is methods of managing comments of music festival visitors.

The aim of the work is to develop software for obtaining information about music festival and communication of its visitors with management means to protect music festival from reputational and economic losses through discriminatory statements of visitors.

Materials, methods and technical means: PHP programming language, Python programming language, GloVe model, Random Forest method.

Results. The result of the work is the developed software and the proposed method. The scientific novelty of the work lies in the proposed method of managing comments of music festival visitors, which is based on the processing of user comments by GloVe model with subsequent classification by Random Forest method, which allows to determine the management decision to avoid reputational losses for the festival.

Conclusions. The developed software allows to combine the possibilities of online communication with the purchase of tickets for music festival to provide appropriate means to avoid problematic visits.

Field of use – medium and large music festivals.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Проблема керування коментарями користувачів	11
1.2 Огляд програмних засобів підтримки відвідувачів музичних фестивалів	13
1.3 Висновки за розділом 1	16
2 Матеріали і методи.....	18
2.1 Модель оброблення природної мови GloVe.....	18
2.2 Моделі класифікації тексту	19
2.3 Метод керування коментарями відвідувачів музичного фестивалю.....	22
2.4 Висновки за розділом 2	25
3 Проєктування програмного забезпечення	26
3.1 Вибір мови програмування	26
3.2 Функціональні вимоги до програмного забезпечення	27
3.3 Проєктування бази даних	32
3.4 Структура програмного забезпечення	37
3.5 Висновки за розділом 3	39
4 Основні рішення щодо реалізації компонентів системи.....	40
4.1 Структура розробленої програмної системи.....	40
4.2 Опис модулів реалізованої програмної системи.....	41
4.3 Висновки за розділом 4	49
5 Експлуатація, тестування та експериментальне дослідження програми	50
5.1 Призначення програми	50
5.2 Умови виконання програми	50
5.3 Виконання програми.....	50
5.4 Експериментальне дослідження методу керування коментарями відвідувачів музичного фестивалю.....	56

5.5 Висновки за розділом 5	57
Висновки	59
Перелік джерел посилання	61
Додаток А Технічне завдання	63
Додаток Б Текст програми	67
Додаток В Слайди презентації.....	88

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

SQL – Structured Query Language;

БД – база даних.

ВСТУП

Суспільство переживає поточний стан розвитку, який характеризується нетерпимістю до дискримінації. Достатньо багато випадків по всьому світу мають відношення до тих чи інших проявів дискримінації і характеризуються активною реакцією суспільства. Така активна реакція в економічній сфері виражається в репутаційних ризиках, які відповідно виражаються в зменшенні прибутків або збитках. Для будь-якої великої компанії важливо відслідковувати будь-які такі прояви і вчасно на них реагувати.

Безсумнівно, що великий музичний фестиваль відноситься до подій, організація яких може постраждати від випадків, пов'язаних з дискримінацією. У той же час організація великого музичного фестивалю – це робота компанії, яка у випадку його проведення з меншою кількістю відвідувачів ніж заплановано або скасування призведе до зменшення фінансових результатів, їх негативного значення або підсумкового банкрутства. Тому для такої компанії важливо вчасно реагувати на інформаційні приводи. Складність фестивалів полягає в тому, що такі інформаційні приводи створювати можуть не тільки самі організатори, але і відвідувачі, люди, які придбали квитки на фестиваль і ще навіть не знаходяться на його території. Кількість відповідних повідомлень від відвідувачів може бути такою, що проаналізувати їх протягом години неможливо, окрім того аналізувати їх 24 години на добу також достатньо складно, відповідно для цього потрібне програмне забезпечення, яке дозволить фактично керувати коментарями відвідувачів музичного фестивалю. Задачею такого програмного забезпечення є виконання аналізу коментарів відвідувачів і визначення рішення за кожним таким коментарем. Якщо він є ризикованим, то бажано його позбутися. При цьому важливо коректно оцінити ризикованість, тобто потрібен достатньо точний механізм аналізу. Окрім того важливо, щоб такий механізм був інтегрований з механізмом продажу квитків на фестиваль, адже це призведе до наявності

впливу на відвідувачів. Цей вплив передовсім потрібен не для того, щоб змусити користувача відмовитись від його думки, а головне для того, щоб за потреби, тобто тоді, коли він часто створює потенційні негативні інформаційні приводи, тобто може ставати активним генератором і передумовою об'єднання інших користувачів, скасувати його відвідування фестивалю. Інтеграція до вебзастосунку, який надає інформацію про фестиваль, механізму продажу квитків, спілкування користувачів, є важливим і актуальним завданням.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Проблема керування коментарями користувачів

Свобода вираження поглядів є важливим правом людини, але також важливим є право не бути дискримінованим, їх потрібно взаємовраховувати:

- ніхто не хоче бути дискримінованим або виключеним через походження, колір шкіри чи релігії – заборона дискримінації є основою прав людини;

- загалом, свобода вираження поглядів супроводжується деякими обмеженнями: не можна випадково розкривати секрети компанії (наприклад, дані клієнтів або виробничий процес) або військові плани уряду, а в умовах надзвичайного стану або війни уряд може обмежити вільне вираження своїх громадян;

- заборонено ображати інших людей або поширювати неправду про них, і людина не повинна підбурювати людей до насильства, дискримінації чи вбивства: якщо це буде зроблено, людина буде покарана законом у більшості країн, тому суд може засудити когось за підбурювання до дискримінації;

- хоча межа між дискримінацією та свободою слова не завжди чітка, існують настанови та угоди, які допомагають її знайти [1].

Мова ворожнечі включає будь-яку форму вираження, включно з текстом, зображенням і відео, при цьому загальне визначення класифікує мову ненависті як будь-яку промову, яка атакує та намагається підпорядкувати будь-яку групу або клас людей, і як правило, висловлюється групою з вищим рівнем соціальної влади, ніж та, на яку направлена промова, цілі таких виступів зазвичай включають расові меншини, жінок, релігійні меншини та представників різних спільнот, такі слова можуть не тільки викликати у людей страждання, страх і приниження, але й призвести до поширення насильницьких емоцій і навіть насильницьких злочинів у суспільстві, беручи до уваги шкоду мови ворожнечі, багато країн і територій

випустили закони, що регулюють мову ворожнечі (уряд Німеччини ухвалив кілька найсуворіших законів щодо мовлення ворожнечі в Інтернеті, наприклад, вимагається від онлайнплатформ видаляти «очевидно незаконні» ворожі висловлювання протягом 24 годин після отримання сповіщення, інакше їх оштрафують) [2].

У статті 10(2) Європейської конвенції з прав людини сказано, що право на свободу вираження поглядів «несе з собою обов'язки та відповідальність», через це право на вільне вираження може підлягати обмеженням та умовам, які необхідні в демократичному суспільстві і чітко викладені, будь-які обмеження мають стосуватися однієї або кількох з наступних цілей: національної безпеки, громадської безпеки, територіальної цілісності, запобігання злочинам чи безладам, захисту здоров'я чи моралі, захисту репутації чи прав інших, запобігання розголошенню інформації, отриманої конфіденційно, та для збереження авторитету та неупередженості судової влади [3].

Виходячи з даного твердження, яке підтримується європейськими законами, музичний фестиваль має право на захист своєї репутації, моралі загалом та прав і репутації інших, тому за цією умовою музичний фестиваль може виконувати керування повідомленнями користувачів, а вебзастосунок повинен організовувати з'єднання потрібних каналів, оскільки музичний фестиваль контактує з власним відвідувачем фактично з моменту придбання квитка (а фактично навіть раніше, коли людина стає користувачем вебсайту музичного фестивалю або учасником його сторінки у соціальній мережі) і продовжує цей процес і під час реалізації фестивалю, коли відвідувач присутній фізично. Тобто загалом це може бути процес тривалістю у рік (така періодичність зазвичай характерна для проведення фестивалів), який переростає після цього у наступний процес тривалістю у рік.

1.2 Огляд програмних засобів підтримки відвідувачів музичних фестивалів

Стандартний вебсайт музичного фестивалю містить основну інформацію про фестиваль. Прикладом такого застосунку розглянуто вебсайт іспанського музичного фестивалю Mad Cool (рис. 1.1) [4].

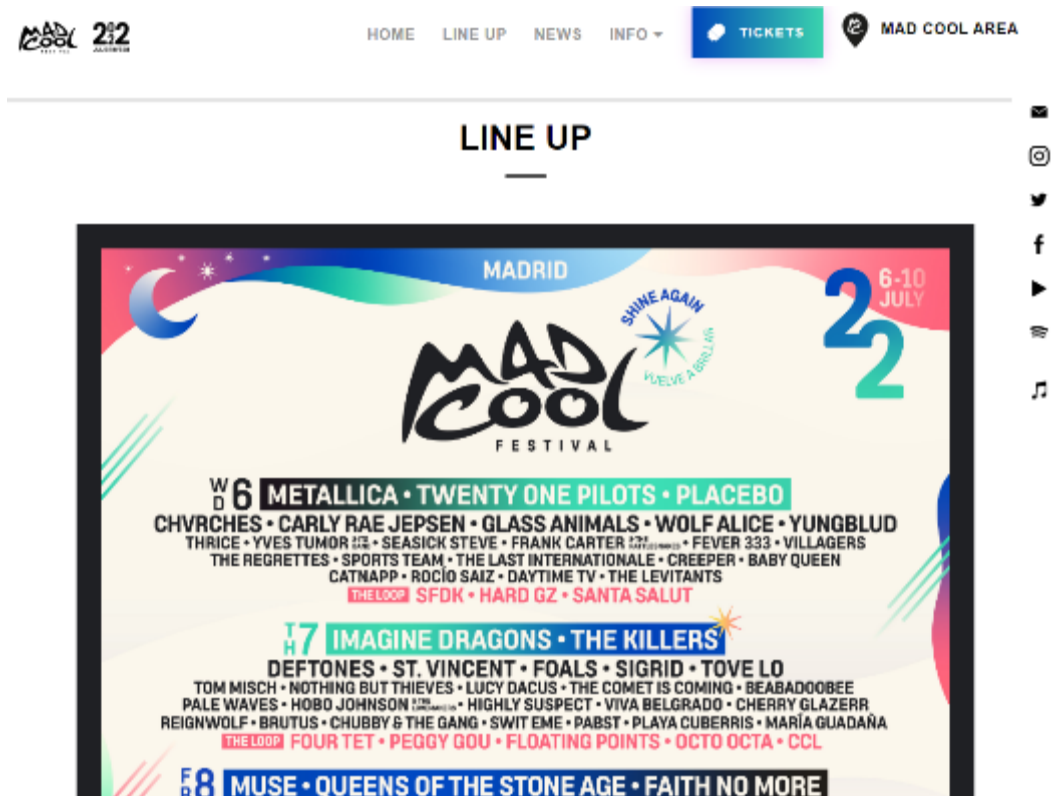


Рисунок 1.1 – Вебсайт музичного фестивалю Mad Cool

Даний вебзастосунок надає інформацію загалом про фестиваль, про розклад виступу артистів, новини, квитки (надаються можливості придбати квитки, зберігати інформацію про них, змінити дату квитків та повернути квитки), запитання, що ставляться часто, зону проведення фестивалю, а також зовнішні ресурси.

Другим прикладом є вебсайт бельгійського фестивалю Rock Werchter (рис. 1.2) [5]. Він надає інформацію про фестиваль, про доступні для придбання квитки, розклад виступу артистів, зону кемпінгу фестивалю,

запитання, що часто ставляться, історію проведення фестивалю, переглянути відео фестивалю, надає можливість виконати пошук по вебсайту.

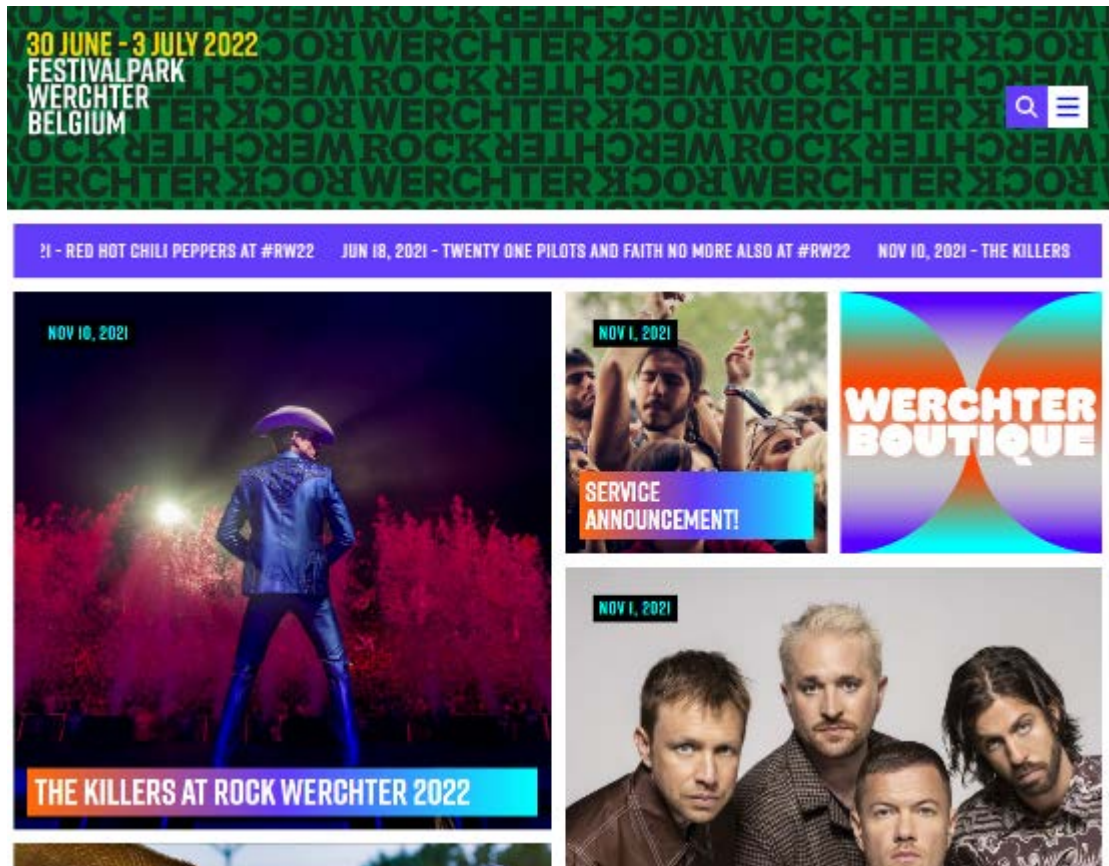


Рисунок 1.2 – Вебсайт музичного фестивалю Rock Werchter

Третім прикладом є вебзастосунок музичного фестивалю Primavera Sound (рис. 1.3) [6]. Він надає набагато ширші засоби взаємодії з користувачем. Фестиваль проходить в різних містах, тому для цього доступні розділи, що відповідають кожному з цих міст, окрім того доступний розділ для перегляду окремих подій в межах сімейства фестивалю (це можуть бути окремі концерти, що презентують фестиваль). За кожним містом доступний розклад виступів його артистів, квитки на даний фестиваль, можливості акредитації, перегляду інформації про фестиваль, запитання, що часто вставляються, інформація про проживання, прибуття на фестиваль.

На загальній сторінці фестивалю доступні розділи з новинами фестивалю, інформацією про фестиваль загалом, інформацією для преси,

контактні дані, засіб пошуку по вебсайту.

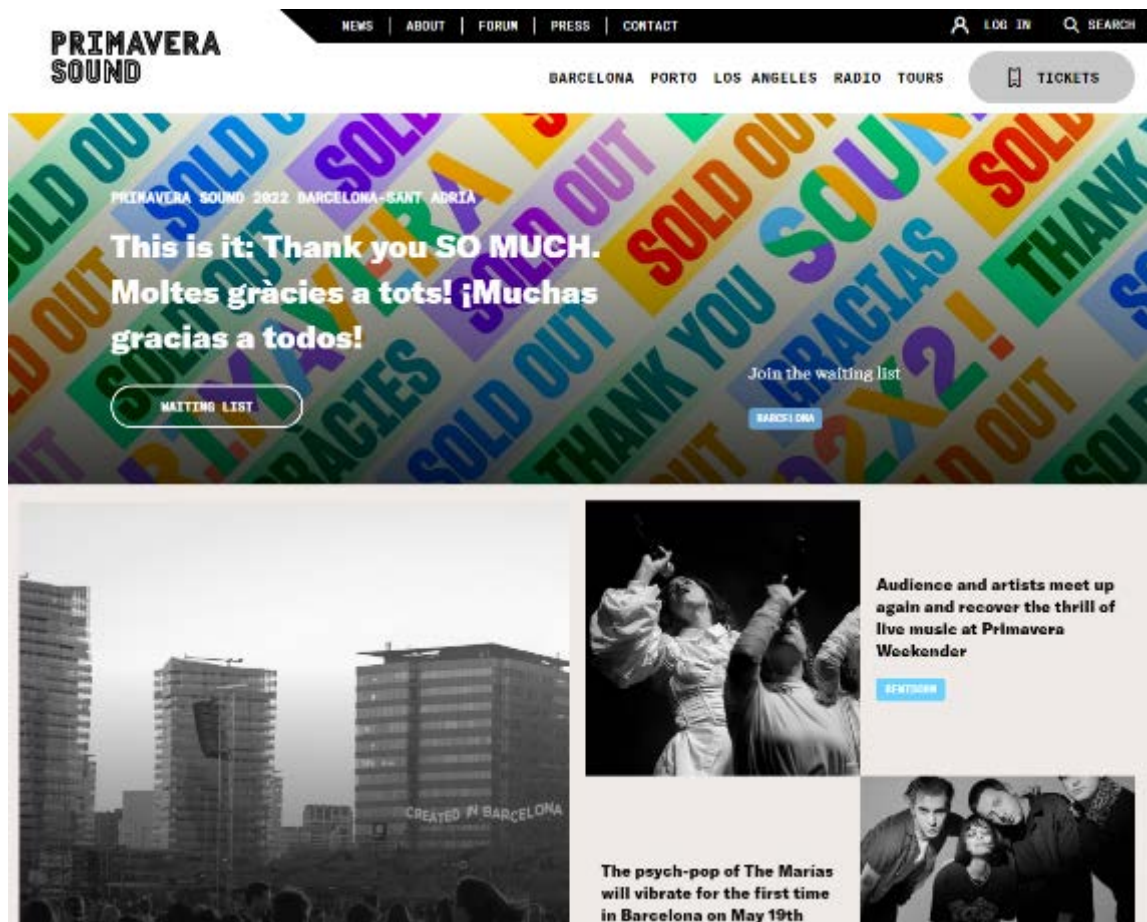


Рисунок 1.3 – Вебсайт музичного фестивалю Primavera Sound

Основною ж особливістю вебсайту є наявність форуму для користувачів. Звичайні користувачі можуть переглядати повідомлення форуму, але опублікувати повідомлення може тільки авторизований користувач. Форум (рис. 1.4) містить набір тем, за кожною з яких фіксується кількість голосів, кількість повідомлень (коментарів), переглядів. За кожною темою надається інформація про дату створення, автора. Це дозволяє створювати спільноту відвідувачів фестивалю, мати для них єдину точку онлайнспілкування, об'єднуючи користувачів з абсолютно різних міст світу. Якщо у фестивалю є тільки сторінка в соціальній мережі для спілкування з відвідувачами, то все їх спілкування буде направлене на ті теми, які публікує фестиваль, тобто це призведе до пошуку ними засобів для спілкування. Так

можуть створюватися окремі канали в соціальних мережах для спілкування, але в такому випадку створюється точка взаємодії користувачів, за яку з одного боку не несе відповідальності фестиваль, але при цьому зовнішні учасники можуть сприймати їх спілкування за рахунок великої кількості учасників (адже іншого ресурсу немає) як презентацію їх позиції, а вже позиція учасників (відвідувачів) асоціюється з позицією фестивалю, що призводить до виникнення репутаційних ризиків.

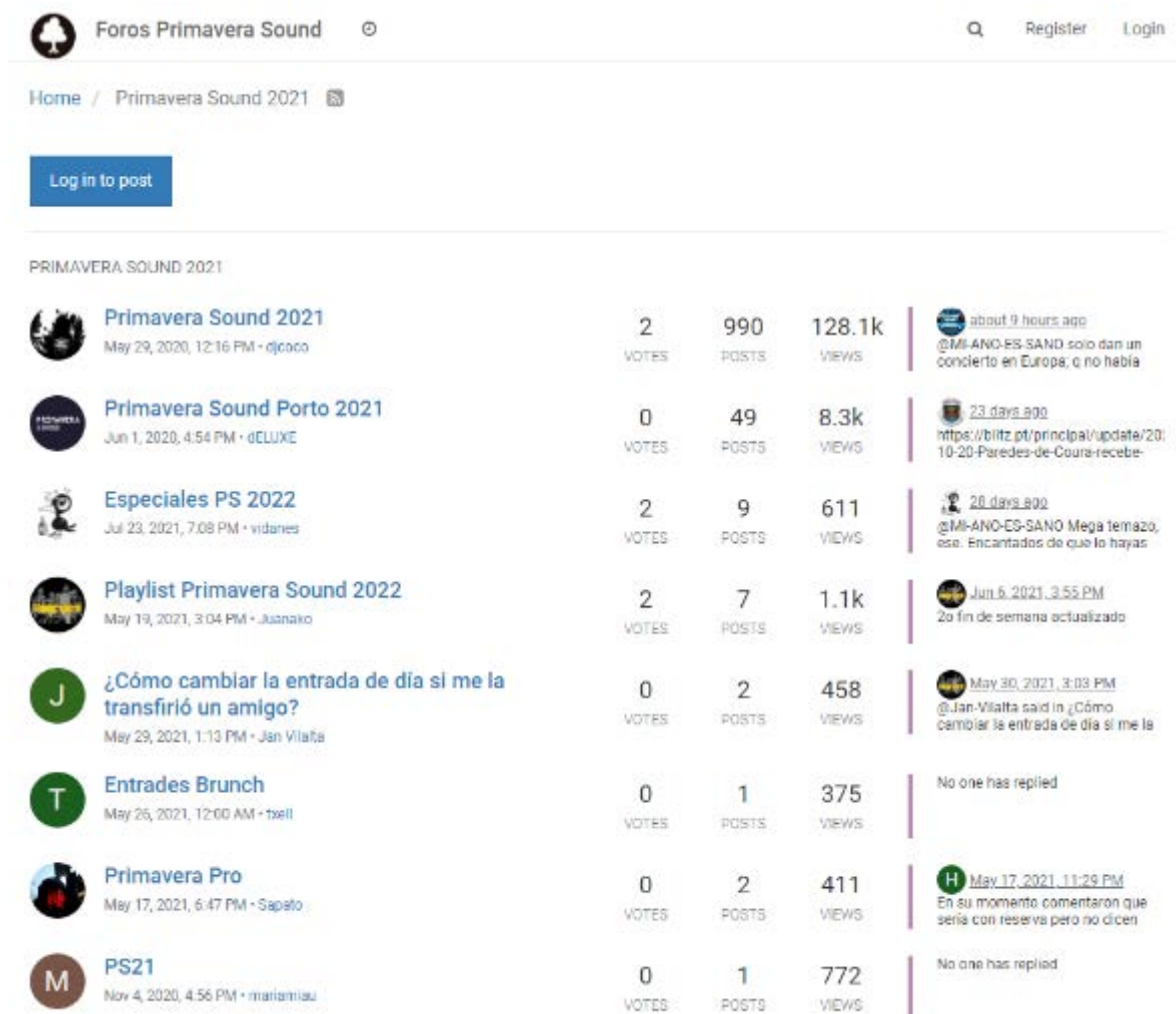


Рисунок 1.4 – Форум музичного фестивалю Primavera Sound

1.3 Висновки за розділом 1

За результатами виконання першого розділу представлено проблему керування коментарями користувачів в розрізі керування коментарями

відвідувачів музичного фестивалю, розглянуто співвідношення свободи вираження думок з дискримінацією людей на основі мови ворожнечі, виконано огляд програмних засобів підтримки відвідувачів музичних фестивалів. За результатами виконаного огляду визначено в якості перспективного засобу для концентрації спілкування відвідувачів музичного фестивалю форум.

У роботі необхідно розв'язати наступні завдання:

- провести дослідження методів керування коментарями відвідувачів музичного фестивалю;
- виконати проєктування програмного забезпечення;
- реалізувати програмне забезпечення.

2 МАТЕРІАЛИ І МЕТОДИ

Для прийняття керуючих рішень щодо впливу на коментар, опублікований користувачем, потрібно виконати класифікацію коментарів, але при цьому слід розуміти, що вхідними даними для такої класифікації є текстові дані, які потребують попередньої обробки, для того щоб їх можна було застосувати в моделі класифікації. Для цього має застосовуватися засіб оброблення природної мови.

2.1 Модель оброблення природної мови GloVe

Вбудовування слова – це закодоване векторне представлення слів, у якому вектори, ближчі один до одного у векторному просторі, подібні за своїм семантичним значенням: використовуючи такі представлення слів, можна створювати моделі для вирішення таких випадків використання тексту в реальному часі, як пошук інформації, класифікація документів, відповіді на запитання, розпізнавання іменованих об'єктів та розбір тексту [7].

GloVe – це модель навчання без учителя, заснована на підрахунку, яка використовує статистику спільної появи (як часто два слова з'являються разом) на глобальному рівні для моделювання векторного представлення слів, оскільки статистика фіксується на глобальному рівні безпосередньо за допомогою моделі, її називають моделлю глобальних векторів, звідки і походить назва [7].

При використанні даної моделі спочатку визначається матриця спільної появи пар слів, кожен елемент якої P_{ij} дорівнює відношенню кількості появ даного слова (першого слова з пари з номером i) у контексті другого слова з номером j (з пари) до кількості появ першого слова загалом.

У такому випадку можна визначити співвідношення між даними оцінками, що складають елементи матриці. За значенням даного співвідношення можна встановити наступне:

- якщо відношення вірогідності появи першого слова в контексті другого до оцінки появи третього слова в контексті другого буде достатньо великим значенням (більше 1), то це означає, що перше слово релевантне другому, а третє – ні;

- якщо відношення вірогідності появи першого слова в контексті другого до оцінки появи третього слова в контексті другого буде достатньо невеликим (менше 1), то це означає, що третє слово релевантне другому, а перше – ні;

- якщо відношення вірогідності появи першого слова в контексті другого до оцінки появи третього слова в контексті другого буде близьким до 1, то це означає, що третє і перше слова однаково релевантні або нерелевантні другому [8].

Модель GloVe навчається на ненульових записах глобальної матриці спільної появи пар слів, яка відображає, як часто слова зустрічаються одне з одним у даному корпусі, при цьому заповнення цієї матриці вимагає одного проходу через весь корпус для збору статистичних даних, для великих корпусів ця дія може бути витратною з точки зору обчислень, але це одноразові початкові витрати, адже подальші ітерації навчання відбуваються набагато швидше, оскільки кількість ненульових записів матриці зазвичай набагато менша за загальну кількість слів у корпусі [9].

2.2 Моделі класифікації тексту

Під час аналізу методів, які можуть використовуватися для класифікації за кодовими представленнями текстових даних, було виділено два основні рішення: метод Random Forest та нейронні мережі. За результатами порівняння було обрано саме Random Forest, оскільки вони

вимагають менших обсягів даних, що дуже важливо для даної проблеми, коли кількість прикладів коментарів з різних класів може змінюватися і розширюватися у випадку виявлення нових суспільних проблем, окрім того вони ще вимагають менших обчислювальних ресурсів, до того ж дозволяють досягати спеціалізації у навчанні.

Метод Random Forest (випадковий ліс) – це алгоритм навчання з учителем, а так званий ліс, який він будує, представляє собою ансамбль дерев рішень, зазвичай навчених методом мішування (bagging): загальна ідея мішування полягає в тому, що комбінація моделей навчання підвищує загальний результат [10].

Особливості алгоритму Random Forest:

- точніший за алгоритм дерева рішень;
- забезпечує ефективний спосіб обробки пропущених даних;
- може давати обґрунтований прогноз без налаштування гіперпараметрів;
- вирішує проблему перенавчання в деревах рішень;
- у кожному випадковому лісовому дереві підмножина ознак вибирається випадковим чином у точці розбиття вузла [11]-[13].

Основними блоками методу Random Forest є дерева рішень, відносно яких можна виділити наступне:

- дерево рішень – це техніка підтримки прийняття рішень, яка утворює деревоподібну структуру;
- дерево рішень складається з трьох компонентів: вузлів рішень, листових вузлів і кореневого вузла;
- алгоритм дерева рішень розділяє навчальний набір даних на гілки, які далі розділяються на інші гілки, ця послідовність триває до тих пір, поки не буде досягнуто листовий вузол, листовий вузол не може бути відокремлений далі;

- вузли в дереві рішень представляють атрибути, які використовуються для визначення результату, вузли рішення забезпечують посилення на листки;
- ентропія та отримання інформації є будівельними блоками дерев рішень;
- ентропія – це метрика для обчислення невизначеності;
- інформаційний приріст – це міра того, як зменшується невизначеність цільової змінної з урахуванням набору незалежних змінних;
- концепція інформаційного приросту передбачає використання незалежних змінних для отримання інформації про цільову змінну (клас): ентропія цільової змінної та умовна ентропія використовуються для оцінки інформаційного приросту, у цьому випадку умовна ентропія віднімається від ентропії цільової змінної;
- інформаційний приріст використовується при навчанні дерев рішень, це допомагає зменшити невизначеність цих дерев: високий приріст інформації означає високий ступінь невизначеності (інформаційна ентропія) була усунена;
- ентропія та інформаційний приріст є важливими при розщепленні гілок, що є важливою операцією при побудові дерев рішень [11].

Основна відмінність між базовим алгоритмом дерев рішень і алгоритмом Random Forest полягає в тому, що встановлення кореневих вузлів і сегрегаційних вузлів в останньому виконується випадковим чином:

- алгоритм Random Forest використовує метод мішування для визначення необхідного результату;
- мішування передбачає використання різних зразків даних (навчальних даних), а не лише одного зразка: набір навчальних даних містить спостереження та функції, які використовуються для визначення результату, дерева рішень видають різні результати залежно від навчальних даних, що подаються до алгоритму, ці результати будуть ранжовані, а найвищий буде обраний як кінцевий результат;

– замість того, щоб мати єдине дерево рішень, випадковий ліс матиме багато дерев рішень: наприклад, маємо чотири дерева рішень, у цьому випадку навчальні дані, що містять спостереження та характеристики об'єктів, будуть розділені на чотири кореневі вузли, кореневі вузли можуть представляти чотири функції, які можуть впливати на вибір клієнта (наприклад, ціна, внутрішня пам'ять, камера та оперативна пам'ять), випадковий ліс розділить вузли шляхом випадкового вибору об'єктів, остаточний прогноз буде обрано на основі результатів чотирьох дерев, результат, обраний більшістю дерев рішень, буде остаточним вибором;

– реалізація класифікації у випадкових лісах виконується наступним чином: для досягнення результату класифікації випадкові ліси використовують методологію ансамблю, навчальні дані подаються для навчання різних дерев рішень, цей набір даних складається із спостережень та ознак, які будуть вибрані випадковим чином під час поділу вузлів, система лісів спирається на різні дерева рішень, листовий вузол кожного дерева є кінцевим результатом, отриманим цим конкретним деревом рішень, відбір остаточного результату здійснюється за мажоритарною системою голосування [11]-[12].

2.3 Метод керування коментарями відвідувачів музичного фестивалю

Результатом керування коментарями відвідувачів музичного фестивалю є співвідношення коментаря з ризикованим або навпаки неризикованим. Окрім того додатковим результатом може бути блокування квитка відвідувача музичного фестивалю. Блокування квитка здійснюється тільки у випадку, якщо користувач певну кількість разів повторно створював ризиковані коментарі, при чому такий ризик був підтверджений адміністратором. Тобто блокування квитка не може бути результатом створення ризикованого коментаря, а є тільки додатковим рішенням до

визнання коментаря ризикованим, при чому тільки в тому випадку, якщо таке рішення виноситься вже не перший раз.

Для визначення стану, коли квиток має бути заблокований, використовується рейтинг користувача. У найпростішому випадку рейтинг користувача може обчислюватися як кількість випадків публікації ризикованих коментарів. Наприклад, спочатку користувачу може надаватися певний позитивний рейтинг, а у подальшому за кожного ризикованого коментаря рейтинг зменшується на 1. Як тільки рейтинг досягає 0 і користувач знову публікує ризикований коментар, квиток може бути заблокований автоматично.

Метод керування коментарями відвідувачів музичного фестивалю має реалізовуватися на основі наступної процедури.

Етап 1. Формування вибірки коментарів відвідувачів музичного фестивалю.

Кожен коментар повинен складатися з текстового повідомлення, довжина якого може бути обмежена, наприклад, в стандартному випадку 255 символами.

Кожен екземпляр вибірки складається з коментаря та класу ризикованості. Коментар має відноситися до класу ризикованих, якщо він містить дискримінаційний вміст стосовно інших людей. У протилежних випадках коментар відноситься до складу неризикованих.

Етап 2. Визначення векторного представлення слів з вибірки коментарів на основі використання моделі GloVe.

Етап 3. Виконання методу Random Forest на основі вибірки коментарів з навчанням випадкового лісу класифікації ризикованості коментарів відвідувачів музичного фестивалю.

Етап 4. Використання випадкового лісу для класифікації нового коментаря та переобчислення рейтингу користувача.

Етап 5. Якщо рейтинг користувача менше мінімального рівня, то заблокувати його квиток для відвідування фестивалю.

Етапи 4-5 повторюються постійно протягом роботи вебзастосунку, як тільки користувач публікує нове повідомлення. Для кожного нового повідомлення реалізується виконання даних етапів.

На рис. 2.1 представлене схематичне відображення роботи методу керування коментарями відвідувачів музичного фестивалю.

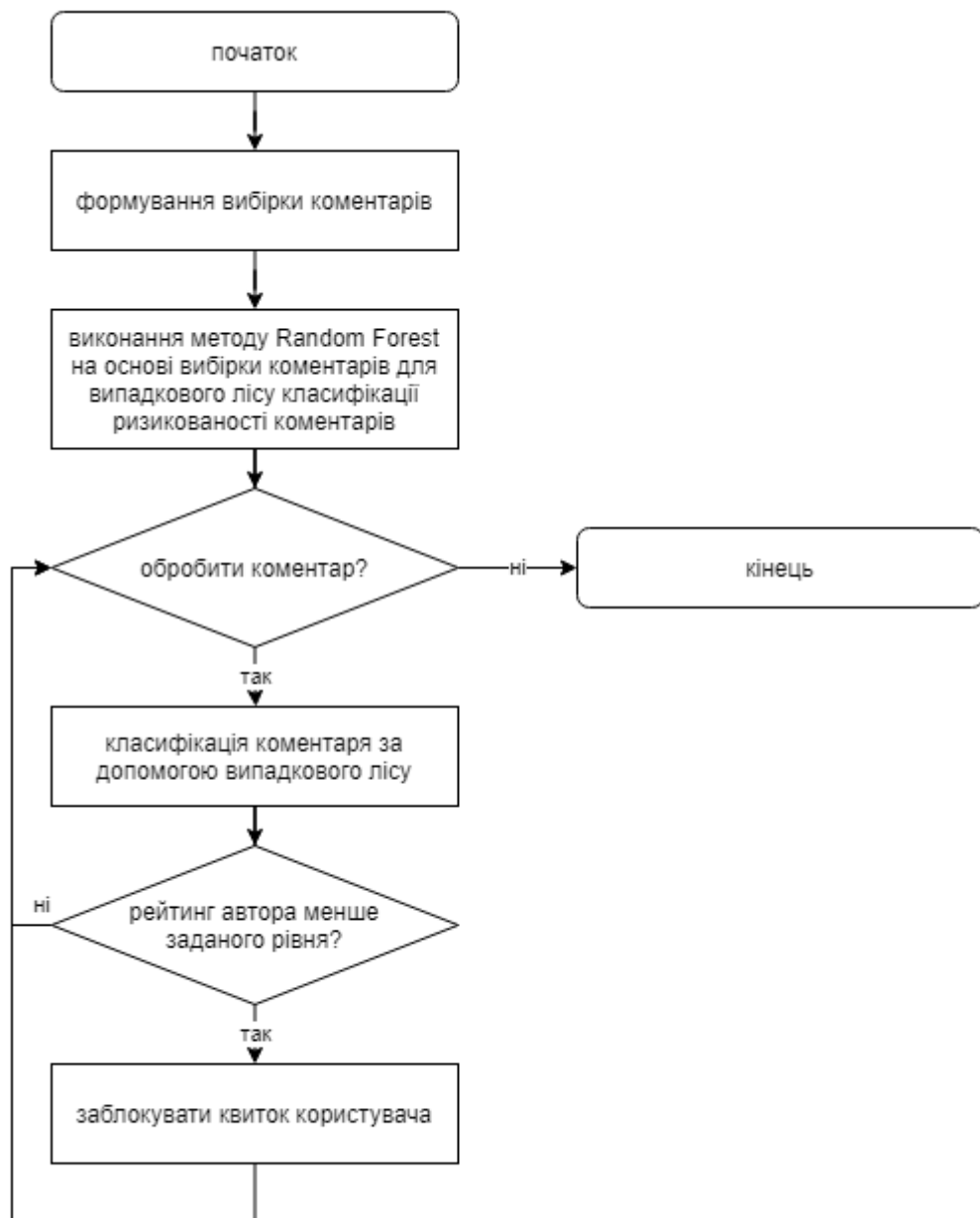


Рисунок 2.1 – Метод керування коментарями відвідувачів музичного фестивалю

2.4 Висновки за розділом 2

За результатами виконання другого розділу визначено потрібні результати виконання керуючого впливу на користувачів, які є відвідувачами музичного фестивалю. Проаналізовано модель оброблення природної мови GloVe та проаналізовано методи, які можуть використовуватися для класифікації коментарів.

Запропоновано власний метод керування коментарями відвідувачів музичного фестивалю, який ґрунтується на виконанні оброблення коментаря користувача моделлю GloVe з подальшою класифікацією методом Random Forest, що дозволяє визначати керуюче рішення з метою уникнення репутаційних втрат для фестивалю.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір мови програмування

Задля забезпечення доступу різних типів користувачів з різних місць (потенційно всього світу), а також потенційно з різних пристроїв, програмне забезпечення необхідно розробити як вебзастосунок. Після прийняття цього рішення було виконано вибір мови програмування для виконання розробки. Альтернативними рішеннями розглянуто мови C# та PHP (табл. 3.1) [14]-[16]. Оскільки мова PHP є відкритим рішенням, незалежним від платформи, потребує менших витрат на хостинг, до того ж характеризується якістю і простотою використання для веброзробки, то саме за її допомогою було вирішено виконувати розробку програмного забезпечення.

Таблиця 3.1 – Порівняння мов програмування

Критерій	C#	PHP
Об'єктно-орієнтована парадигма програмування	Так	Так
Кількість розробників	Менша	Більша
Відкритість програмного забезпечення	Ні	Так
Незалежність від платформи	Неповна, розвивається	Так
Хостинг	Більш коштовний	Менш коштовний
Підтримка веброзробки	Так	Так

Реалізацію методу керування коментарями відвідувачів музичного фестивалю має бути виконано окремо на основі мови програмування Python, адже вона з легкістю дозволяє реалізувати роботу з моделлю GloVe та методом Random Forest, що значно зменшить потрібний час [17]-[18].

3.2 Функціональні вимоги до програмного забезпечення

Для забезпечення взаємодії з користувачем-відвідувачем музичного фестивалю програмне забезпечення, що розробляється, повинно відповідати наступним функціональним вимогам:

- створення тем для спілкування на форумі фестивалю;
- публікація коментарів в темі форуму;
- уподобання коментарів;
- невподобання коментарів;
- відображення коментарів у темі форуму;
- відображення тем форуму;
- голосування за тему форуму;
- відслідковування переглядів теми форуму;
- пошук тем форуму;
- відображення розкладу виступу артистів;
- придбання квитків;
- заборона на доступ користувачу;
- виявлення повідомлень з репутаційним ризиком;
- визначення і відображення артистів;
- визначення часу і місця виступу артистів;
- визначення сцен фестивалю;
- скасування квитка;
- створення і відображення персонального розкладу відвідування фестивалю;
- уподобання виступу артиста;
- відображення власних голосів за теми форуму;
- відображення оцінок власних коментарів;
- відображення оцінених користувачем коментарів;
- відображення квитків користувача;
- визначення груп тарифів квитків;

- визначення вартісних пропозицій для груп квитків;
- відображення варіантів придбання квитків;
- відображення і керування заблокованими користувачами;
- відображення і керування скасованими квитками;
- відображення і керування повідомленнями з репутаційним ризиком;
- авторизація і реєстрація користувачів.

Базово з програмою працює неавторизований користувач. Програма для нього має надавати загальну інформацію про фестиваль, що в результаті має перетворити його за згоди у авторизованого користувача або відвідувача фестивалю.

На рис. 3.1 сценарії було промодельовано за допомогою діаграми варіантів використання програмного забезпечення неавторизованим користувачем.

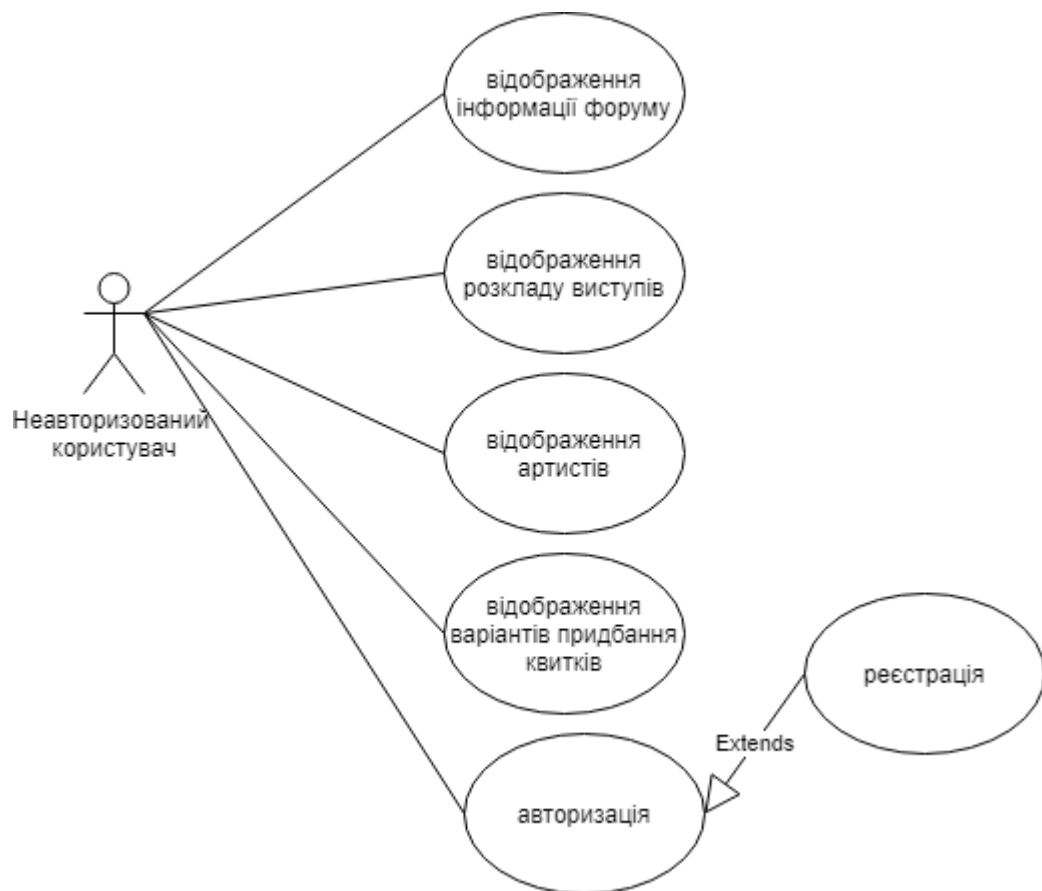


Рисунок 3.1 – Діаграма варіантів використання програмного забезпечення неавторизованим користувачем

Відповідно сценарії роботи неавторизованого користувача включають:

- відображення інформації форуму;
- відображення розкладу виступів;
- відображення артистів;
- відображення варіантів придбання квитків;
- авторизація і реєстрація користувачів.

Перелік сценаріїв авторизованого користувача (рис. 3.2):

- голосування за тему форуму;
- придбання квитків;
- відображення власних активностей;
- уподобання коментаря;
- публікація коментарів в темі форуму.

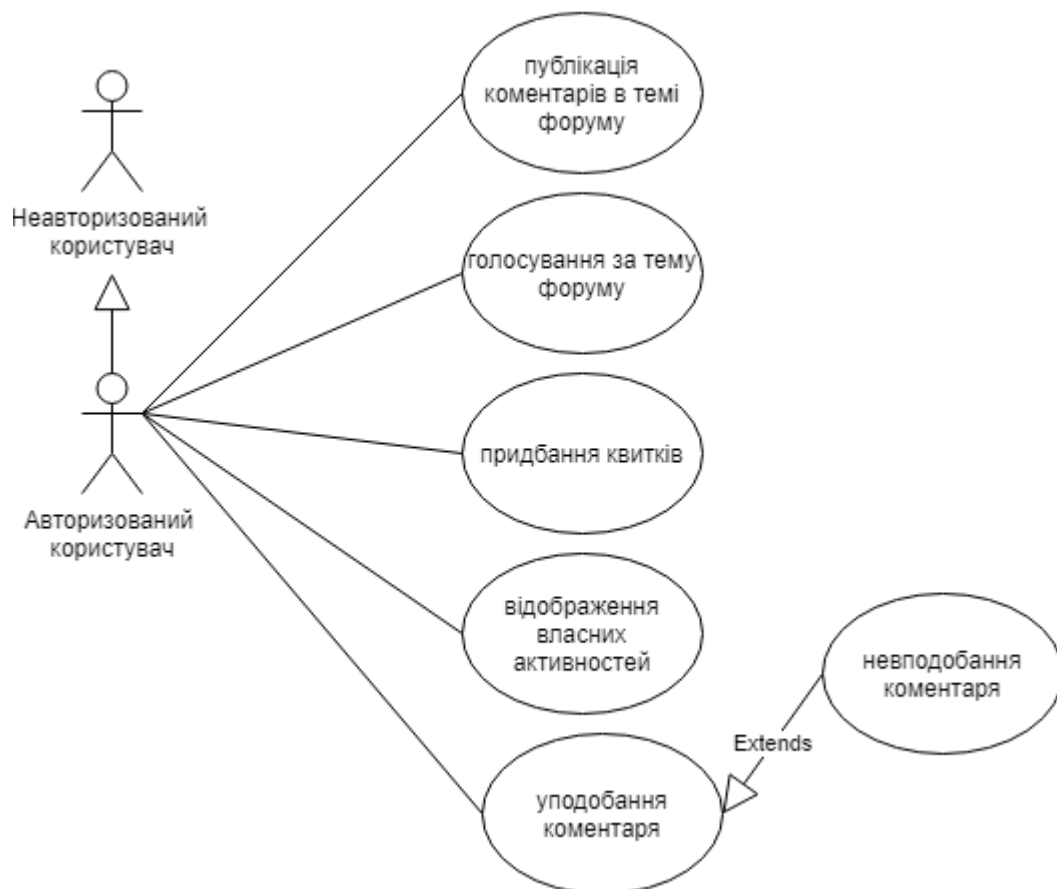


Рисунок 3.2 – Діаграма варіантів використання програмного забезпечення авторизованим користувачем

Відображення власних активностей включає відображення голосів за теми форуму, відображення власних коментарів і оцінок власних коментарів, відображення оцінених користувачем коментарів.

Відвідувачем фестивалю може бути авторизований користувач, тому що тільки авторизованому користувачу доступна можливість придбання квитків. Перелік сценаріїв відвідувача фестивалю (рис. 3.3):

- створення тем для спілкування на форумі фестивалю;
- створення і відображення персонального розкладу відвідування фестивалю;
- відображення квитків користувача.

Адміністратор може керувати загальними даними про музичний фестиваль, а також може керувати можливостями інших користувачів.

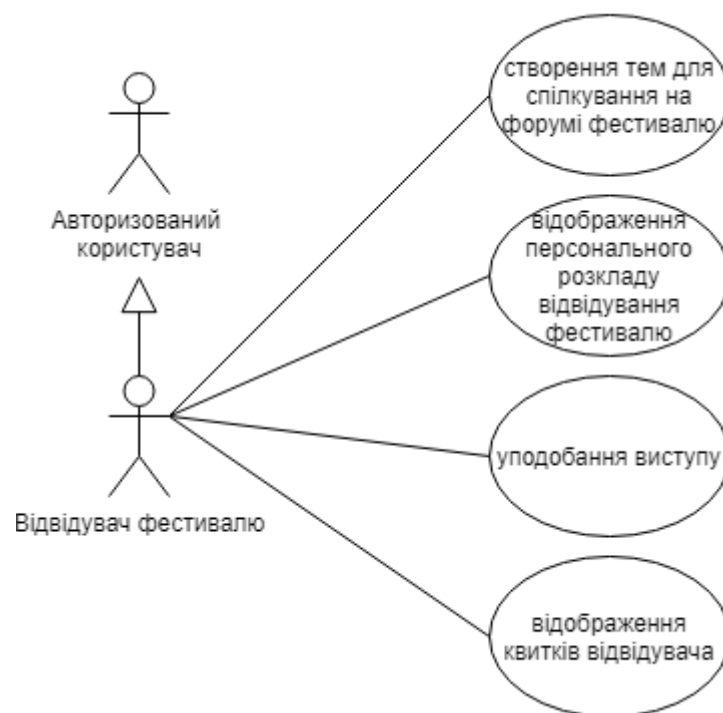


Рисунок 3.3 – Діаграма варіантів використання програмного забезпечення відвідувачем фестивалю

До сценаріїв адміністратора (рис. 3.4) належать:

- керування темами форуму;
- керування інформацією про артистів;

- керування розкладом виступів артистів;
- керування сценами фестивалю.



Рисунок 3.4 – Діаграма варіантів використання програмного забезпечення адміністратором

До сценаріїв адміністратора також належать:

- авторизація адміністратора;
- керування доступом користувачів;
- керування повідомленнями з репутаційним ризиком;
- керування квитками;
- керування групами тарифів.

3.3 Проєктування бази даних

У даному підрозділі представлено структуру розробленої бази даних (БД). Використано систему керування БД MySQL.

Таблиця Artist представляє сутність, що визначає особу або колектив, які виступають на музичному фестивалі, і включає поля:

- id – номер артиста;
- name – назва (ім'я) артиста;
- country – країна, яку представляє артист;
- description – основна інформація про артиста;
- site – вебсайт артиста;
- ymsite – сторінка артиста на сервісі Youtube Music;
- spotsite – сторінка артиста на сервісі Spotify;
- fbpage – сторінка артиста в соціальній мережі Facebook;
- twpage – сторінка артиста в соціальній мережі Twitter;
- inpage – сторінка артиста в соціальній мережі Instagram;
- youpage – сторінка артиста на сервісі Youtube;
- youvideo – відео презентації артиста (відео з його Youtube-сторінки);
- level – цифрове значення рівня артиста серед всіх (порядок у переліку).

Сутність сцена представляє сцену, на якій відбувається виступ на фестивалі. Таблиця Stage включає поля:

- id – номер сцени;

- name – назва сцени;
- description – опис сцени.

Таблиця LineUp визначає розклад виступів артистів на сценах музичного фестивалю і включає поля:

- id – номер виступу;
- artistId – артист, який виступає на сцені музичного фестивалю (зовнішній ключ з таблиці Artist);
- stageId – сцена, на якій виступає артист (зовнішній ключ з таблиці Stage);
- begin – початок виступу;
- end – закінчення виступу.

Оскільки користувач на основі уподобань артистів створює свій власний розклад, то було виділено відповідну сутність, що визначила таблицю Watch, яка включає поля:

- id – номер запланованого перегляду виступу артиста;
- lineupId – виступ артиста (зовнішній ключ з таблиці LineUp);
- userId – користувач (зовнішній ключ з таблиці User);
- when – час, коли було виконано планування.

Таблиця User визначає сутність, що відповідає користувачу вебзастосунку, і включає поля:

- id – номер користувача;
- name – ім'я користувача;
- login – логін користувача;
- password – пароль користувача;
- rating – рейтинг користувача;
- role – роль користувача;
- phone – номер телефону користувача;
- photo – фотографія користувача;
- state – стан користувача.

Таблиця Topic представляє сутність, що визначає тему загального форуму музичного фестивалю, де спілкуються між собою користувачів, і включає поля:

- id – номер теми форуму;
- name – назва теми;
- userId – користувач, який створив тему (зовнішній ключ з таблиці User);
- votes – кількість голосів за дану тему форуму;
- views – кількість переглядів теми;
- when – час створення теми форуму.

Таблиця Ban визначає заборону на роботу користувача з програмою і включає поля:

- id – номер заборони;
- userId – номер користувача, для якого визначено заборону (зовнішній ключ з таблиці User);
- from – час початку дії заборони;
- to – час завершення дії заборони;
- state – стан заборони.

Таблиця View визначає перегляд користувачем теми форуму і включає поля:

- id – номер перегляду теми форуму;
- userId – номер користувача, який переглядав тему форуму (зовнішній ключ з таблиці User);
- topicId – номер теми форуму, яку було переглянуто (зовнішній ключ з таблиці Topic);
- when – час перегляду.

Таблиця Vote визначає голоси, віддані користувачем за тему форуму, і включає поля:

- id – номер голосу за тему форуму;
- userId – користувач, автор голосу (зовнішній ключ з таблиці User);

- topicId – тема форуму, за яку було віддано голос (зовнішній ключ з таблиці Topic);

- when – час голосування.

Таблиця Liker визначає уподобання та неуподобання користувачем коментарів і включає поля:

- id – номер уподобання коментаря;

- commentId – номер уподобаного коментаря (зовнішній ключ з таблиці Comment);

- userId – номер користувача (зовнішній ключ з таблиці User);

- when – час уподобання;

- type – тип уподобання (уподобання або навпаки невподоба).

Таблиця TicketGroup визначає групи тарифів на продаж квитків і включає поля:

- id – номер групи квитків;

- from – час початку дії квитка;

- to – час завершення дії квитка;

- category – категорія квитка (необхідна для виділення, наприклад, квитків з доступом до додаткових сервісів);

- quantity – максимальна кількість даної групи квитків, що може бути продана;

- description – опис групи квитків.

Таблиця TicketPrice (рис. 3.5) визначає саме вартісну пропозицію для конкретної групи тарифів і включає поля:

- id – номер вартісної пропозиції на придбання квитків;

- ticketgroupId – група квитків, на яку поширюється дія пропозиції (зовнішній ключ з таблиці TicketGroup);

- price – вартість одного квитка;

- from – початок дії пропозиції;

- to – завершення дії пропозиції.

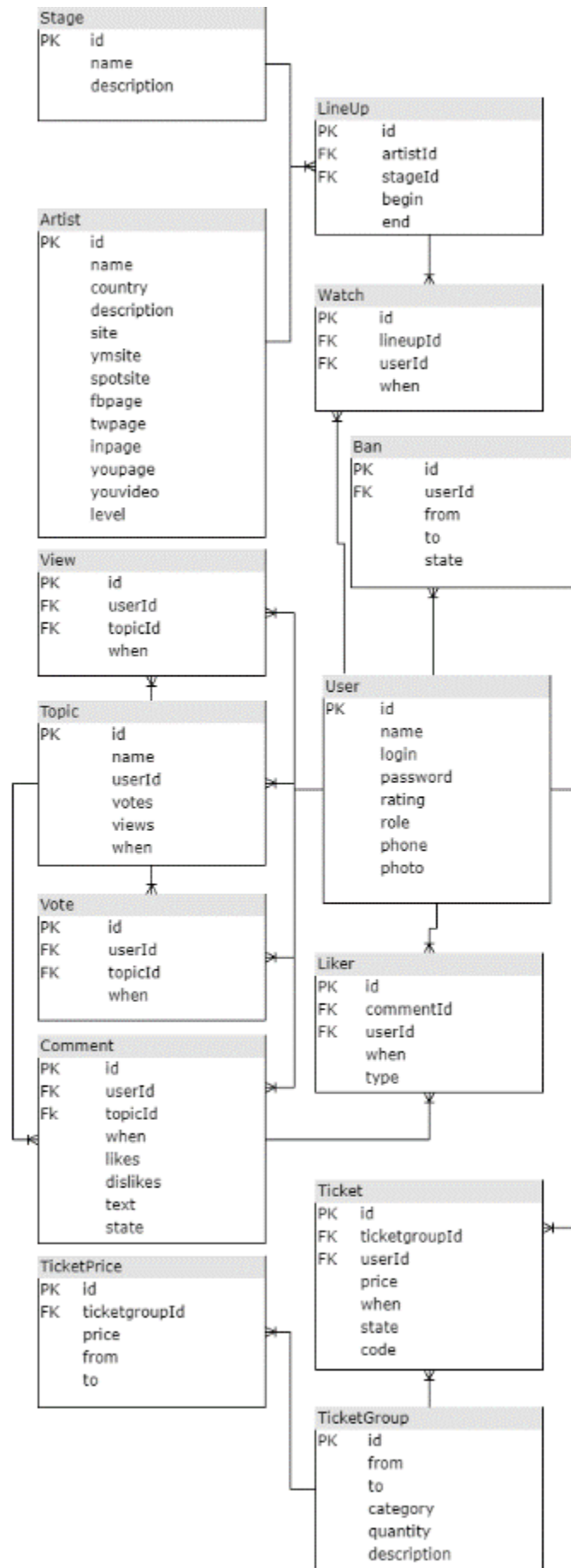


Рисунок 3.5 – Схема БД

Таблиця Ticket визначає конкретний квиток, який дозволяє відвідати музичний фестиваль, і включає поля:

- id – номер квитка;
- ticketgroupId – група квитків, до якої належить даний квиток (зовнішній ключ з таблиці TicketGroup);
- userId – користувач, який придбав квиток (зовнішній ключ з таблиці User);
- price – вартість квитка;
- when – час, коли було придбано квиток;
- state – стан квитка (необхідний для анулювання квитка);
- code – захисний код квитка.

Таблиця Comment визначає коментарі користувачів, які є складовою тем форуму, і включає поля:

- id – номер коментаря;
- userId – номер користувача, який є автором коментаря (зовнішній ключ з таблиці User);
- topicId – номер теми форуму, в якій було розміщено коментар (зовнішній ключ з таблиці Topic);
- when – час розміщення коментаря;
- likes – кількість уподобань коментаря;
- dislikes – кількість неуподобань коментаря;
- text – текст коментаря;
- state – стан коментаря.

3.4 Структура програмного забезпечення

Розроблена структура програмного забезпечення (рис. 3.6) складається з наступних базових одиниць:

- model – включає моделі предметної області;

- controller – включає контролери, що забезпечують підтримку роботи відповідних типів користувачів з програмою;
- template – містить файли стилів, зображень, необхідних для роботи програми, скрипти;
- components – містить базові компоненти, що забезпечують з'єднання з БД DBMFest.php, реалізацію запропонованого методу CommManMethod.py;
- view – містить представлення програми, що включають різні вебсторінки, необхідні для відображення даних користувачу.

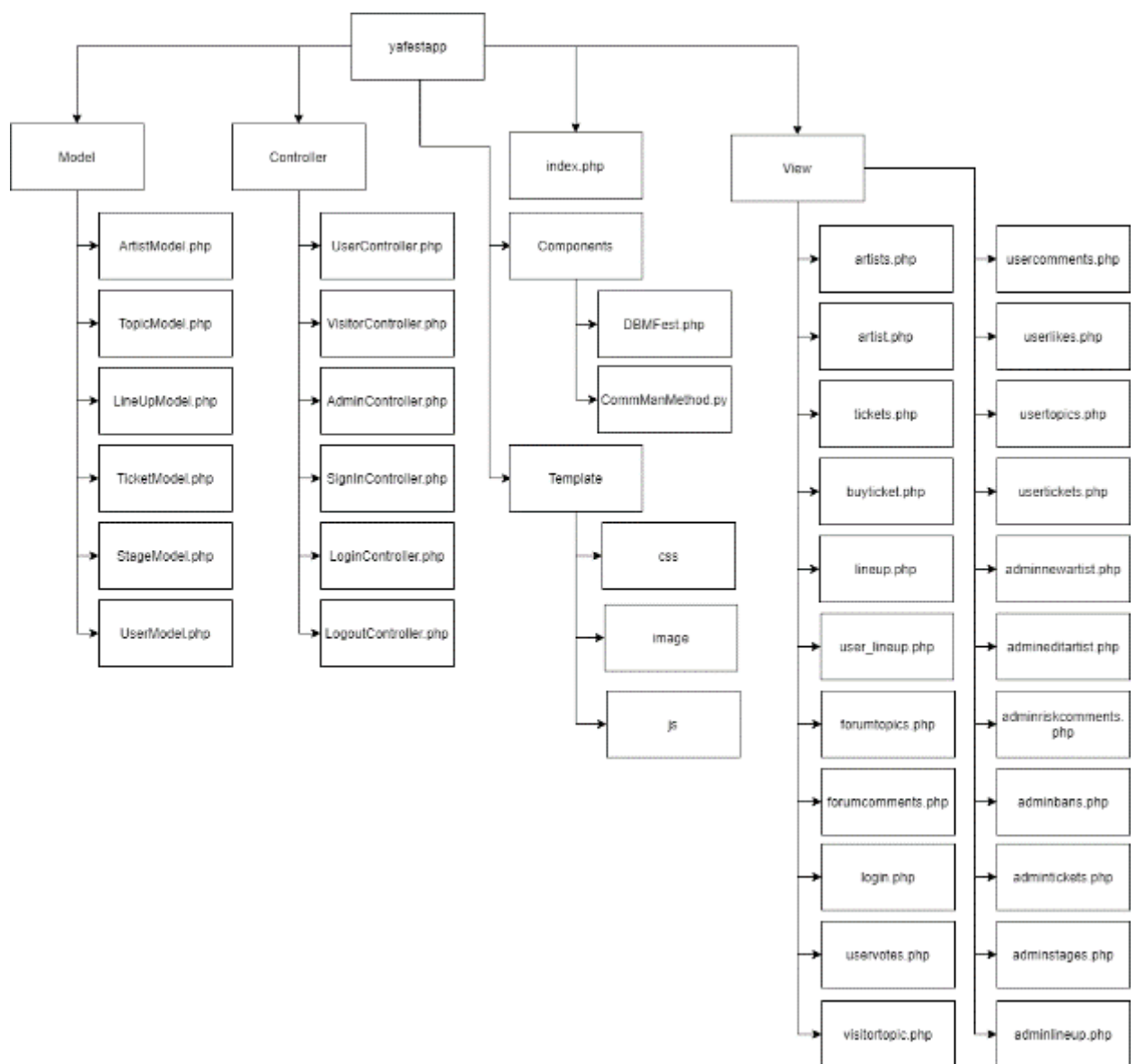


Рисунок 3.6 – Структурна схема програми

3.5 Висновки за розділом 3

За результатами виконання другого розділу обґрунтовано вибір мови програмування PHP для реалізації вебзастосунку, мови програмування Python для реалізації запропонованого методу, визначено функціональні вимоги до програмного забезпечення з моделюванням вимог на основі діаграм варіантів використання, виконано проєктування БД зі створенням її структури, визначено структуру програми.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Структура розробленої програмної системи

Основною сутністю, за допомогою якої реалізується об'єкт роботи, є коментар відвідувача музичного фестивалю. Саме його аналіз здійснюється автоматично програмою і на основі даного аналізу приймається відповідне управлінське рішення. На рис. 4.1 представлена відповідна діаграма станів коментаря відвідувача музичного фестивалю, що демонструє логіку взаємодії між користувачем та програмним забезпеченням відносно реалізації даної основної сутності.

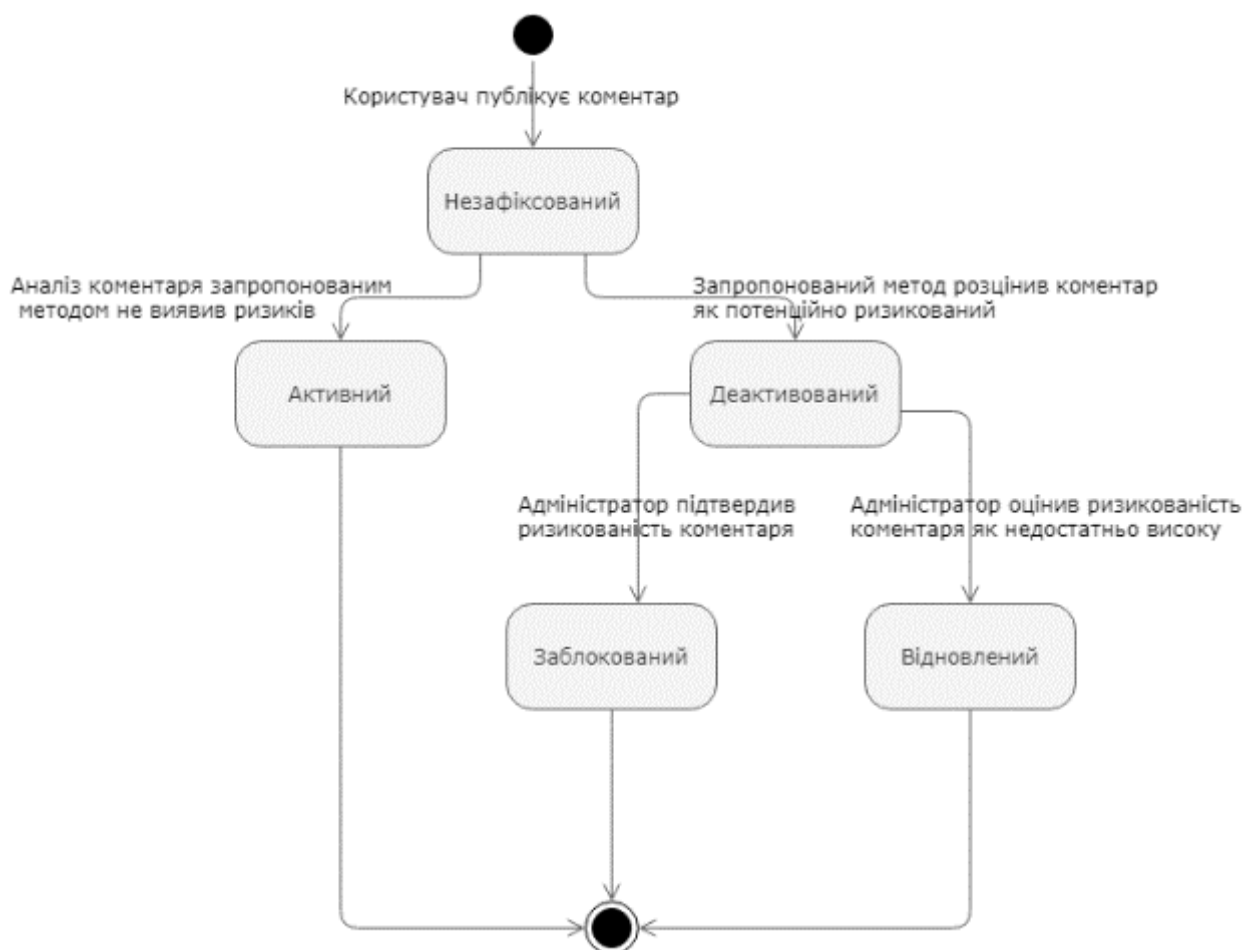


Рисунок 4.1 – Діаграма станів коментаря відвідувача музичного фестивалю

Після створення коментаря користувачем він знаходиться в незафіксованому стані, тобто ще не є збереженим у БД. Після цього коментар піддається аналізу шляхом використання запропонованого методу керування коментарями відвідувачів музичного фестивалю. За результатами такого аналізу має бути винесено одно з двох рішень. Це автоматизовані рішення.

Таким рішенням може бути переведення коментаря до стану активного. У такому випадку коментар буде відображатися на форумі усім користувачам.

Другим рішенням може бути переведення коментаря до стану деактивованого. Це відбувається у випадку, якщо коментар оцінюється як ризикований. Тоді коментар не відображається користувачам при перегляді відповідної теми форуму.

У випадку другого винесеного рішення подальшу підтримку може виконати адміністратор. Він може підтвердити вручну винесене рішення про деактивацію коментаря, якщо погоджується з таким автоматичним рішенням. Тоді коментар переходить до стану заблокованого. Окрім того адміністратор може відхилити автоматичне рішення і перевести коментар до стану відновленого.

4.2 Опис модулів реалізованої програмної системи

В основі реалізації предметної області лежать класи, які було створено для реалізації відповідних моделей. Основні з цих класів будуть описані в даному підрозділі.

Клас `ArtistModel` реалізує роботу над сутністю артиста і включає до свого складу засоби роботи з даними відповідних об'єктів. Робота з даними реалізується шляхом взаємодії з БД на основі виконання відповідних `Structured Query Language (SQL)`-запитів.

Метод `add ($name, $country, $level, $description, $site, $ymsite, $spotsite, $fbpage, $inpage, $youpage, $youvideo)` вносить інформацію про

нового артиста, який має ім'я name, походить з країни country, знаходиться в переліку артистів фестивалю на рівні level (необхідно для виділення хедлайнерів фестивалю, відомих артистів та невідомих), включає інформацію description про себе, має вебсайт site, сторінку на Youtube Music ymsite, сторінку на Spotify spotsite, сторінку Facebook fbpage, Instagram inpage, профіль Youtube youpage та відео, виділене для презентації артиста, youvideo.

Метод edit (\$id, \$name, \$country, \$level, \$description, \$site, \$ymsite, \$spotsite, \$fbpage, \$inpage, \$youpage, \$youvideo) виконує редагування інформації про артиста з номером id, змінюючи його ім'я на name, країну походження на country, рівень у переліку артистів на level, опис на description, вебсайт на site, сторінку на Youtube Music на ymsite, сторінку на Spotify на spotsite, сторінку Facebook на fbpage, Instagram на inpage, профіль Youtube на youpage та відео, виділене для презентації артиста, на youvideo.

Метод getInformation (\$id) повертає інформацію за всіма характеристиками (аналогічно описаним для попередніх методів) про артиста з номером id.

Метод getArtists () повертає перелік усіх артистів, які на даний момент внесені до переліку артистів фестивалю. За кожним артистом з переліку повертається його номер, ім'я, країна, опис, посилання на відео та посилання на вебсайт.

Для формування переліку артистів створюється SQL-запит:

```
$sql = "
    SELECT id, name, country, description, youvideo, site
    FROM Artist
    ORDER BY name
";
```

Потім даний запит виконується шляхом звернення до БД і витягається перелік артистів:

```
$result = $dbc->query($sql);
```

```
$artists = $result->fetchAll(PDO::FETCH_ASSOC);
```

Метод `searchArtist ($name)` виконує пошук артистів, ім'я або назва яких включає в якості своєї частини `name` або дорівнює `name`.

Метод `getArtistsLevel ($level)` повертає перелік усіх артистів, які належать до рівня `level`.

Клас `LineUpModel` забезпечує підтримку роботи з розкладом виступів артистів, надаючи зокрема для цього засоби створення персонального розкладу.

Метод `add ($artist, $stage, $date1, $date2)` вносить дані про виступ артиста з номером `artist` на сцені з номером `stage`, починаючи з моменту `date1` і закінчуючи моментом часу `date2`.

Метод `edit ($id, $artist, $stage, $date1, $date2)` виконує редагування даних про виступ з номером `id`, вносячи в якості дати початку виступу `date1`, в якості дати завершення виступу `date2`, артиста – `artist`, сцени – `stage`.

Метод `lineupDates ()` повертає перелік дат, у які будуть відбуватися виступи на фестивалі.

Метод `lineupStageDate ($stage, $date)` повертає перелік артистів, які мають виступ на сцені `stage` в дату `date`, сортуючи виступи за збільшенням часу.

Метод `lineupStage ($stage, $date)` повертає перелік артистів, які мають виступ на сцені `stage`, сортуючи виступи за збільшенням дати і часу.

Метод `addWatch ($us, $lineup)` вносить виступ артиста під номером `lineup` до персонального розкладу виступів користувача `us`.

Метод `getDateWatches ($us, $date)` повертає власний розклад для користувача з номером `us` на дату `date`.

Метод `getWatches ($us)` повертає власний розклад для користувача з номером `us`.

Клас `TopicModel` реалізує забезпечення засобів роботи над форумом користувачів. Для цього підтримуються засоби роботи над темами форуму, повідомленнями, які розташовуються в темах.

Метод `addTopic ($name, $userId)` виконує додавання теми з назвою `name` в форум за авторством користувача `userId`. Для цього спочатку реалізується перевірка, чи є даний користувач відвідувачем фестивалю. Для цього виконується вибірка з БД для перевірки наявності у нього квитка. Якщо такий квиток у користувача є, при чому цей квиток є активним, то виконується запит до БД на створення теми форуму за авторством даного користувача.

Метод `canAddTopic ($userId)` перевіряє, чи може користувач з номером `userId` створювати у поточний момент тему в форумі, тобто чи є він відвідувачем фестивалю.

Метод `getTopicComments ($topic)` повертає перелік коментарів форуму в темі з номером `topic`, сортуючи їх за часом додавання, залишаючи при цьому тільки активні повідомлення.

Метод `getOwnComments ($id)` повертає перелік усіх коментарів, відсортованих від поточного моменту часу, які були створені користувачем з номером `id`.

Метод `getCommentState ($state)` перетворює числове значення стану коментаря `state` на словесний опис даного стану.

Метод `getTopicsByDate ($up)` повертає перелік усіх тем форуму, відсортованих за датою їх створення, починаючи поточною датою (якщо передано в якості параметра значення `false`) або завершуючи поточною датою (якщо передано значення `true`). За кожною темою повертається її номер, назва, кількість голосів, переглядів, дата і час створення теми, логін і фото користувача, який створив цю тему.

Метод `getTopicsByVotes ($up)` повертає перелік усіх тем форуму, відсортованих за кількістю голосів, відданих за тему, починаючи з найбільшої, якщо передано в якості параметра значення `false`, і у

протилежному напрямку, якщо `up` дорівнює `true`. Перелік характеристик по кожній темі аналогічний попередньому методу.

Метод `editTopic ($id, $tname)` змінює назву теми форуму з номером `id` на `tname`.

Метод `getTopicsByViews ($up)` повертає перелік усіх тем форуму, відсортованих за кількістю переглядів коментарів теми, починаючи з найбільшої, якщо передано в якості параметра значення `false`, і у протилежному напрямку, якщо `up` дорівнює `true`. За кожною темою повертається її номер, назва, кількість голосів, переглядів, дата і час створення теми, логін і фото користувача, який створив цю тему.

Метод `searchTopic ($title)` виконує пошук тем форуму з назвою, що включає в якості своєї частини або дорівнює `title`. Перелік характеристик по кожній темі аналогічний попередньому методу.

Метод `voteTopic ($us, $topic)` забезпечує голосування користувача з номером `us` за тему форуму з номером `topic`, для чого реалізується збереження відповідного голосу та збільшення кількості голосів, відданих за відповідну тему, на 1.

Метод `viewTopic ($us, $topic)` забезпечує внесення даних про перегляд користувачем з номером `us` теми форуму з номером `topic`, для чого реалізується збереження відповідного перегляду та збільшення кількості переглядів відповідної теми на 1.

Метод `getOwnTopics ($id)` повертає перелік тем форуму, які були створені користувачем з номером `id`. За кожною темою повертається її номер, назва, кількість голосів, переглядів, дата і час створення теми.

Метод `getOwnVotes ($us)` повертає перелік голосів за теми форуму, які були віддані користувачем з номером `us`.

Метод `unVoteTopic ($us, $topic)` скасовує голосування користувачем з номером `us` за тему з номером `topic`. Кількість голосів за тему при цьому зменшується на 1.

Метод `likeComment ($us, $comm, $type)` забезпечує уподобання або неуподобання коментаря з номером `comm` користувачем з номером `us`. Під неуподобанням розуміється висловлення негативної думки про коментар. Для висловлення негативної думки потрібно передати в якості значення параметра `type` значення `false`, відповідно для висловлення позитивної думки – `true`. Метод також реалізує перевірку коректності затребуваної дії. Тобто якщо коментар вже був уподобаний, то уподобання буде знято. Якщо коментарю було створене неуподобання і потрібно знову ж виконати цю дію, то неуподобання також буде знято. Якщо потрібно уподобати коментар, а він до цього був неуподобаний, то неуподобання буде знято, а уподобання додано. Якщо потрібно виконати неуподобання коментаря, а він до цього був уподобаний, то уподобання буде знято, а неуподобання додано.

Метод `editComment ($id, $text)` виконує редагування коментаря з номером `id`, змінюючи його текст на `text`.

Метод `getLikedComments ($us, $type)` повертає уподобані користувачем з номером `us` коментарі, якщо в якості значення другого параметра передано `true`, і повертає коментарі, щодо яких цей користувач висловив негативну думку, якщо в якості другого параметра передано значення `false`. За кожною позицією повертається інформація про уподобання, про сам коментар (номер, текст, номер теми форуму).

Метод `getRiskedComments ($state)` повертає перелік ризикованих коментарів, тобто коментарів щодо яких здійснюється автоматичне або ручне керування. Коментарі відсортовані, починаючи з поточного моменту, тобто останній доданий коментар буде першим у переліку. Якщо в якості значення параметра передано `0`, то повертаються деактивовані коментарі, тобто ті, які були автоматично віднесені до цензурованих і приховані в результаті. Якщо значенням параметра є `2`, то повертаються заблоковані коментарі. Якщо значенням параметра є `1`, то повертаються відновлені коментарі.

Метод `addComment ($text, $userId, $topicId)` виконує додавання коментаря з текстом `text` від користувача `userId` до форуму з внесенням до

теми `topicId`. Перед додаванням коментаря виконується перевірка, чи не має користувач з номером `userId` поточну заборону на доступ, що реалізується за допомогою наступного SQL-запиту:

```
$sql = $dbc->prepare("SELECT id, userId, from, to, state
FROM Ban
WHERE userId = :userId AND state = 1 AND from > :today AND to <
:today");
```

Після перевірки заборони перед внесенням коментаря реалізується автоматична перевірка ризикованості повідомлення шляхом звернення до засобів класу `CommentManager`:

```
$state = CommentManager::getDecision($text);
$sql = "INSERT INTO Comment(userId, topicId, likes, dislikes, text, state,
when)
VALUES (:userId, :topicId, 0, 0, :text, :state, :today)";
$result = $dbc->prepare($sql);
```

Окрім того у випадку ризикованості повідомлення рейтинг користувача зменшується. За потреби можна за допомогою відповідного методу змінити рейтинг користувача на потрібний, тобто змінити систему обчислення рейтингу.

Метод `editUserRating ($us, $rating)` змінює рейтинг користувача з номером `us` на значення `rating`.

Метод `cancelBan ($us)` скасовує поточну заборону на доступ користувача з номером `us`. Даний метод дозволяє скасувати тільки активну заборону, тобто якщо строк дії заборони вже минув, то скасувати її неможливо.

Метод `banUser ($us, $from, $to)` створює заборону для користувача з номером `us` на час з `from` до `to`. Заборона діє на додавання нових коментарів у темах форуму.

Метод `approveComment ($id, $state)` забезпечує керування коментарем з номером `id`, який було автоматично віднесено до ризикованих. Якщо в якості значення параметра `state` передано `false`, то коментар буде відновлено. Якщо передано `true`, то коментар буде остаточно заблоковано. Даний метод підтримує роботу адміністратора, який аналізує ризиковані коментарі на форумі і або підтверджує цей ризик для фестивалю, або скасовує його.

Клас `TicketModel` реалізує засоби роботи з квитками на музичний фестиваль.

Метод `cancelticket ($ticket)` скасовує квиток на фестиваль з номером `$ticket`.

Метод `usertickets ($us)` повертає перелік квитків користувача з номером `$us`.

Метод `checkticket ($ticket, $code)` дозволяє перевірити, чи відповідає код `$code` квитку з номером `$ticket`.

Метод `getticket ($ticket)` повертає повну інформацію про квиток з номером `$ticket`.

Метод `getticketgroups ()` повертає перелік доступних на даний момент тарифів на купівлю квитків.

Метод `getticketgroup ($tg)` повертає всю інформацію про групу квитків з номером `$tg`.

Метод `addticket ($us, $category, $from, $to)` вносить інформацію про купівлю користувачем з номером `$us` квитка категорії `$category` на дати з `$from` до `$to`, генеруючи при цьому код для квитка на основі інформації про квиток `$ticket` та генерації випадкової частини.

Метод `$addticketgroup ($from, $to, $category, $quantity, $description)` вносить дані про наявність групи квитків на дати з `$from` до `$to` категорії `$category` у кількості `$quantity` з описом `$description`.

Метод `addticketprice ($tg, $price, $from, $to)` вносить інформацію про тарифну пропозицію за групою квитків `$tg` у розмірі `$price` на дати з `$from` до `$to`.

Представлення організовано з використанням безкоштовного шаблону [19].

4.3 Висновки за розділом 4

Розглянуто можливі стани основної сутності предметної області – коментаря відвідувача музичного фестивалю. Представлено відповідну діаграму станів.

Представлено основні розроблені класи, які реалізують файлову структуру, що була запроєктована в розділі 2. Описано логіку реалізації їх основних методів.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Програма призначена для надання користувачам, які є потенційними відвідувачами фестивалю, інформації про музичний фестиваль, створення майданчика для комунікації відвідувачів фестивалю з можливістю керування ризикованим спілкуванням, отримання власних даних, пов'язаних з фестивалем.

5.2 Умови виконання програми

Апаратне забезпечення роботи вебсервера повинно відповідати вимогам роботи з щонайменше 200 користувачами одночасно. Для підтримки цих вимог апаратне забезпечення повинно мінімально включати:

- процесор: чотири ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять: 10 Гб;
- жорсткий диск: 300 Мб доступного простору.

5.3 Виконання програми

За початкового звернення до вебсервера за адресою вебзастосунку відображається вебсторінка з інформацією про фестиваль (рис. 5.1). Дана сторінка відображається також в меню за пунктом «Про фестиваль».

Окрім даного пункту до меню входять:

- пункт меню «Артисти», що дозволяє переглянути перелік артистів з градацією за рівнями, звідки можна перейти до перегляду детальної інформації про артиста;

– пункт меню «Квитки», який дозволяє переглянути тарифи на придбання квитків та придбати квиток, але безпосередньо перед придбанням потрібно авторизуватися;

– пункт меню «Розклад», який дозволяє переглянути інформацію про час виступу артистів на сценах фестивалю (для цього потрібно обрати день виступу та сцену, а відображається тоді розклад за часом);

– пункт меню «Форум», який дозволяє отримати доступ до спілкування користувачів за темами форуму;

– пункт меню «Увійти», який дозволяє авторизуватися користувачу.

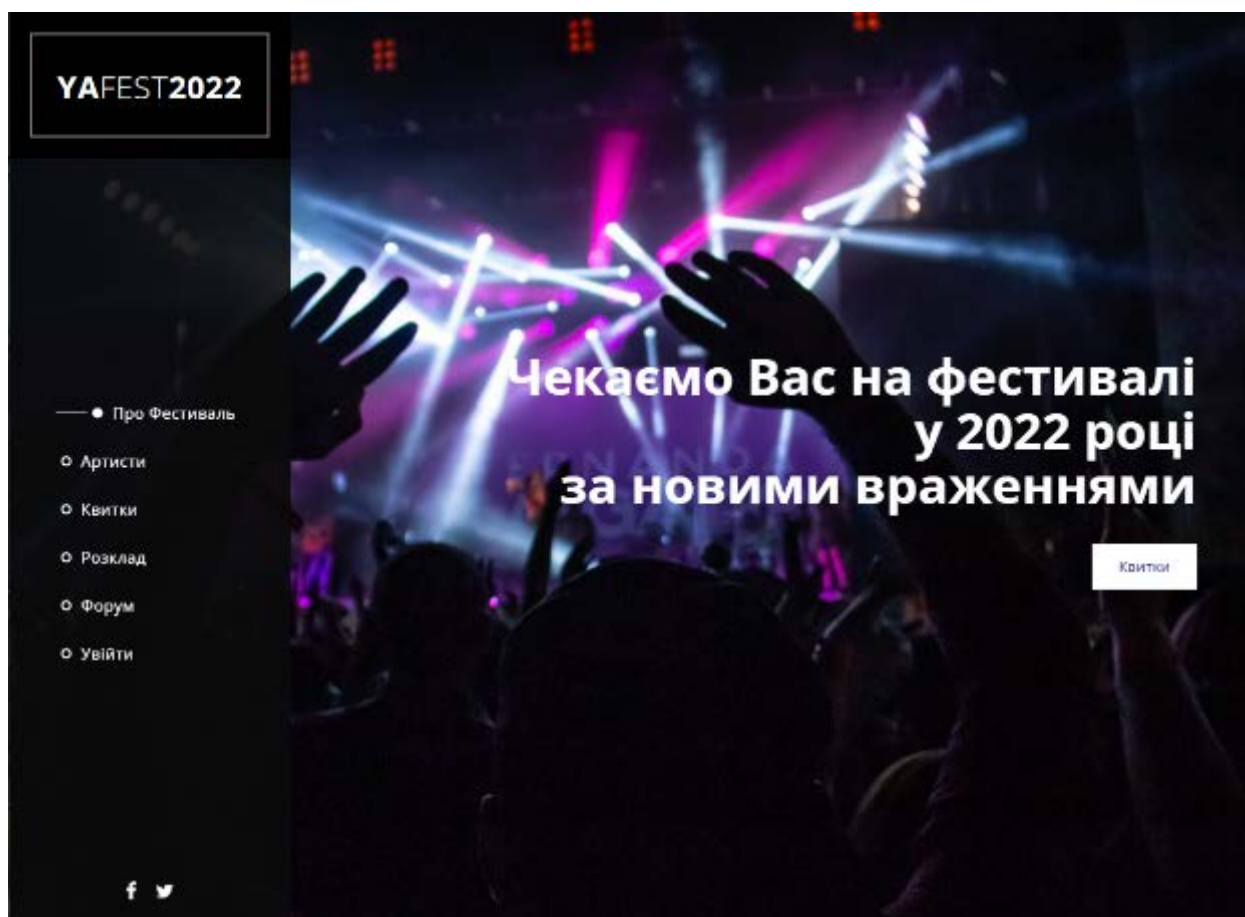


Рисунок 5.1 – Вебсторінка про фестиваль

На рис. 5.2 приведено приклад відображення інформації про обраного з вебсторінки «Артисти» артиста. За ним відображається його назва або ім'я, опис, кнопка «В Гості», що дозволяє перейти на вебсайт артиста, основне

відео з youtube, а також кнопки для переходу на ресурси, що представляють соціальні мережі або зовнішні музичні ресурси.

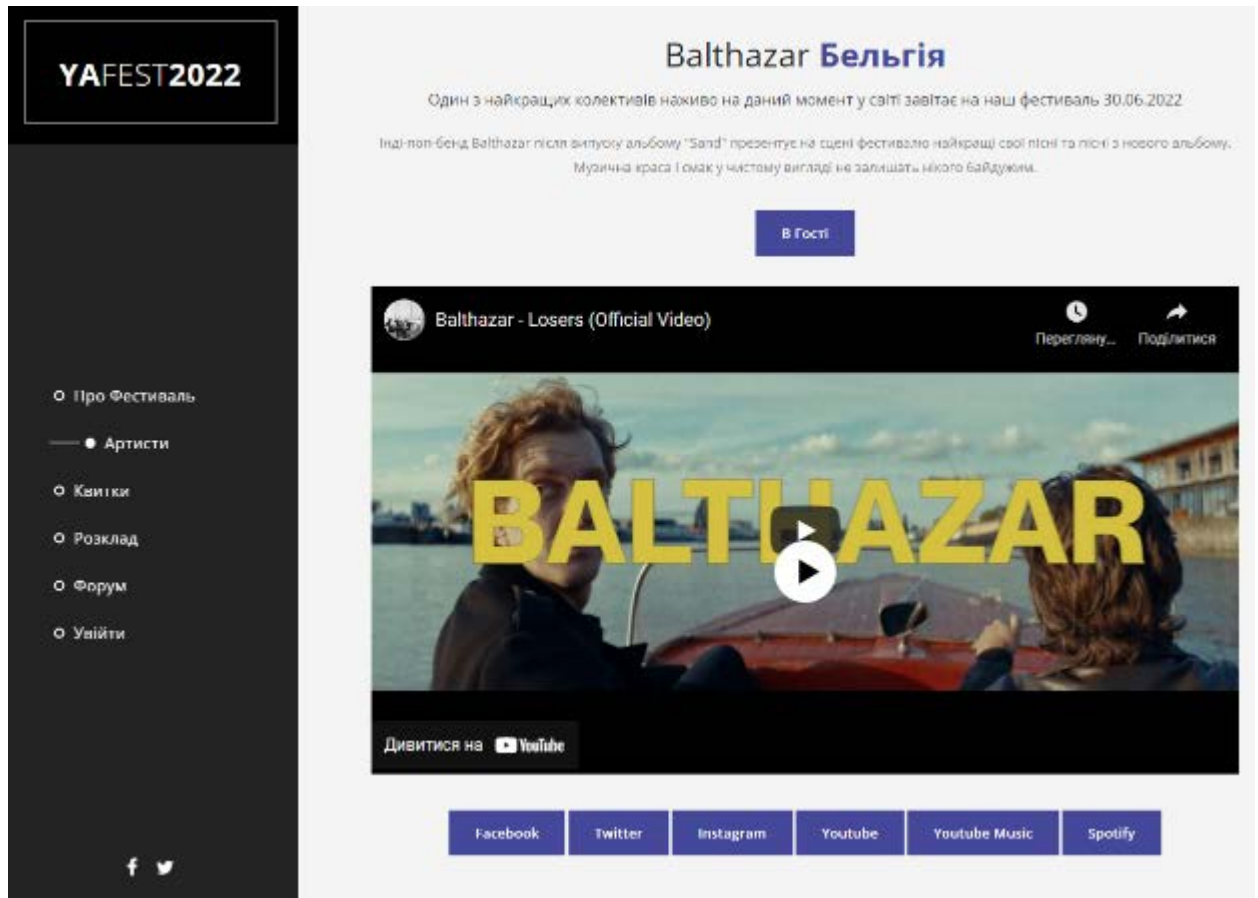


Рисунок 5.2 – Вебсторінка з інформацією про артиста

Якщо користувач авторизувався, то додатково до даних пунктів меню йому доступні наступні.

Пункт меню «Відвідування» дозволяє переглянути власний розклад відвідування виступів артистів (рис. 5.3). Дана вебсторінка має структуру аналогічну вебсторінці «Розклад». У відвідуваннях користувач має обрати потрібну йому дату фестивалю, після чого за часом початку виступу буде відображено виступи артистів, які він вподобав з вебсторінки «Розклад». Якщо користувач змінив свої плани, то він може натиснути на кнопку «Вилучити» поруч з відповідним виступом і позбутися його у своєму розкладі відвідувань. Дана вебсторінка дозволяє спланувати виступи, які хоче відвідати користувач, щоб не пропустити їх на фестивалі.

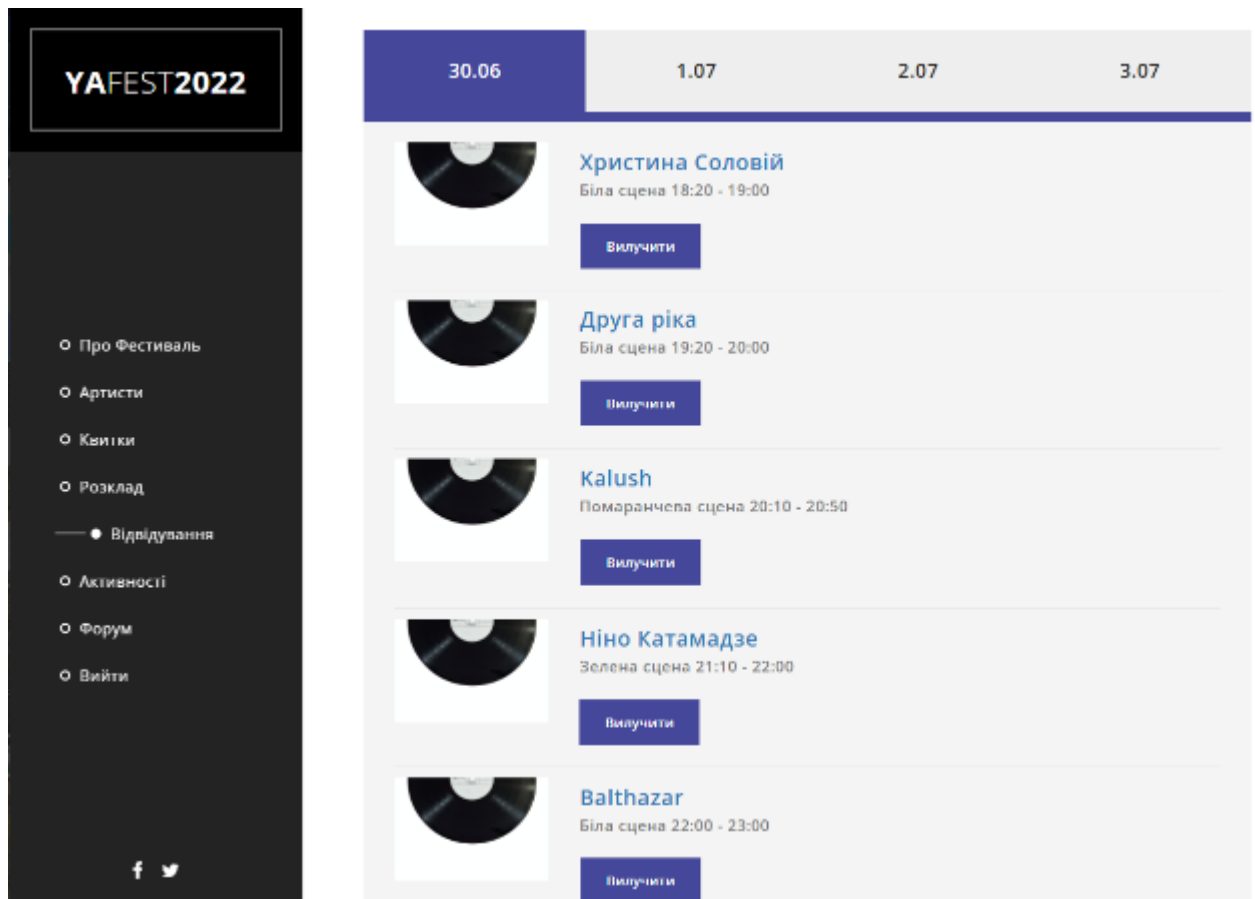


Рисунок 5.3 – Вебсторінка персонального розкладу відвідування фестивалю

Вебсторінка «Активності» дозволяє за подібною системою вибору переглянути «Голоси», звідки відкривається перелік тем форуму, за які проголосував користувач (можна звідси скасувати свій голос), «Коментарі», звідки можна переглянути власні опубліковані коментарі, «Оцінки», звідки можна переглянути виставлені користувачем уподобання або неуподобання коментарів інших користувачів, «Профіль», звідки доступна інформація про користувача. Для відвідувача музичного фестивалю, тобто авторизованого користувача, який має придбаний активний квиток на фестиваль, доступні також на даній сторінці додаткові пункти меню: «Теми», звідки можна переглянути власні створені теми форуму, а також можна створити нову тему форуму, «Квитки», звідки можна переглянути перелік придбаний відвідувачем квитків на фестиваль.

Вебсторінка «Форум» надає доступ до переліку тем форуму, надаючи можливість упорядкувати їх за кількістю голосів або за алфавітом. Після

відкриття окремої теми (рис. 5.4) відкривається вебсторінка, де відображається назва теми форуму, її автор, кількість відданих голосів за тему, кнопка «Голосувати», яка дозволяє проголосувати за дану тему. Окрім того виводиться поле для введення власного коментаря користувача, який можна опублікувати, натискаючи на кнопку «Відправити».

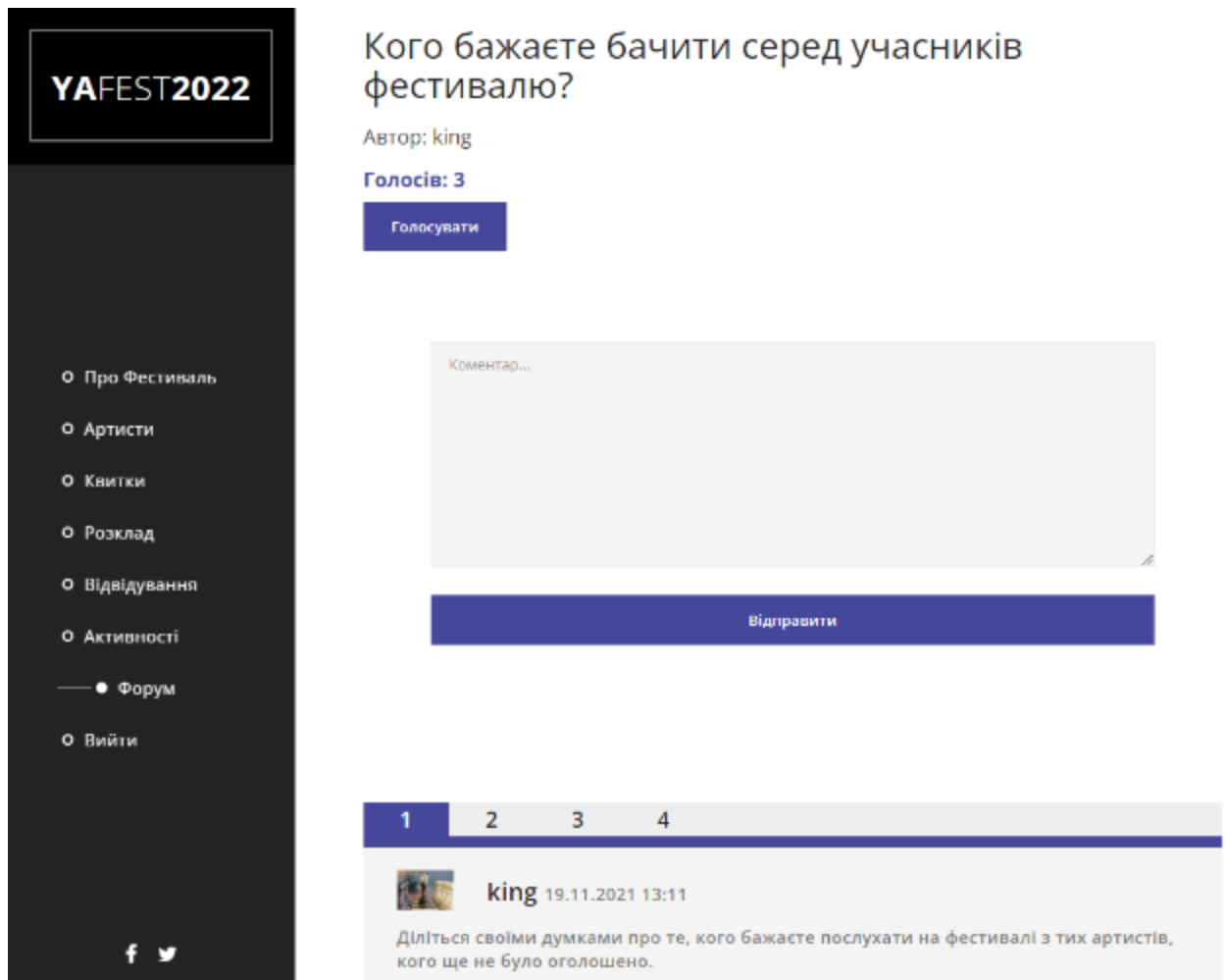


Рисунок 5.4 – Вебсторінка теми форуму

Нижче на сторінці розташовується перелік коментарів. Коментарі розділяються порціями по 10 на окремі сторінці (рис. 5.5). Доступ до номерів сторінок організовано через їх вибір за допомогою системи закладок. Після вибору відповідного номера нижче відображається перелік коментарів. За кожним коментарем відображається зображення користувача, його логін, дата і час публікації коментаря, текст коментаря, кількість уподобань. Окрім

того користувач може уподобати коментар або вказати, що коментар йому не подобається за допомогою відповідних посилань.

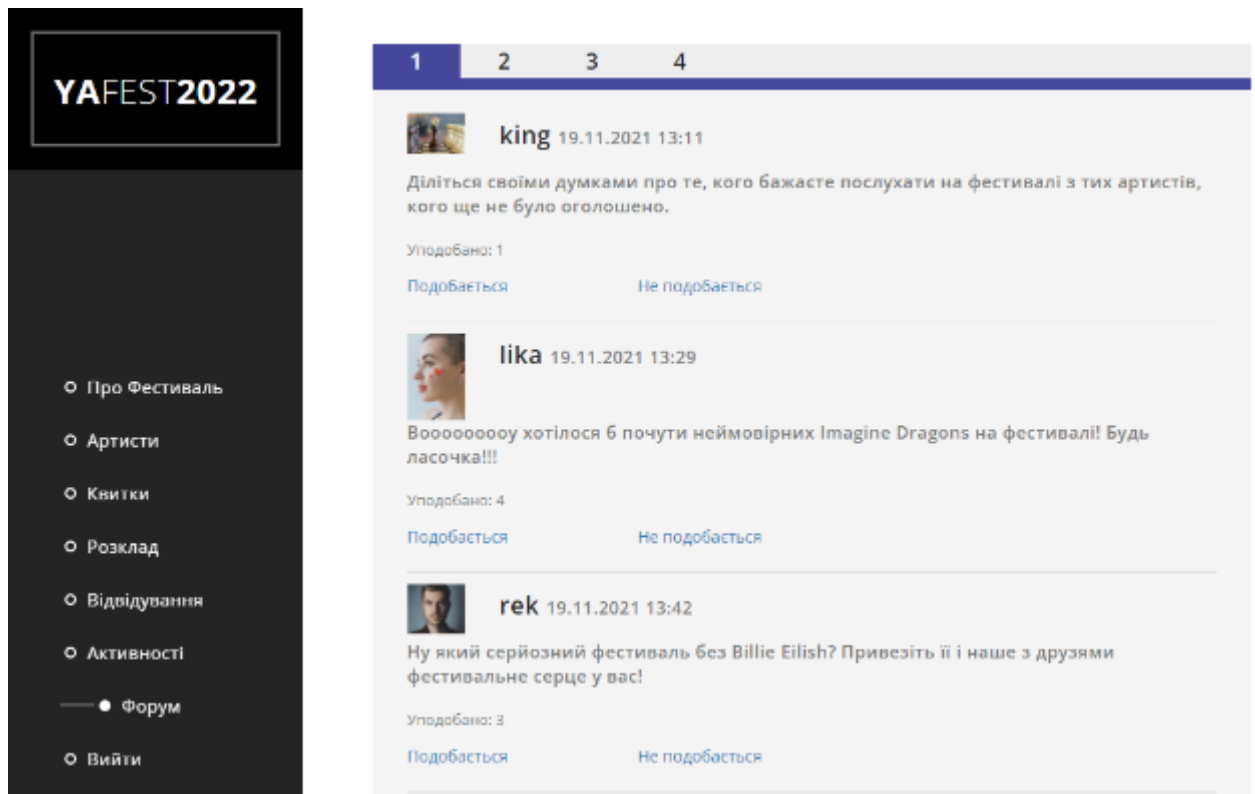


Рисунок 5.5 – Вебсторінка теми форуму з коментарями

Адміністратору доступно меню, в якому є пункт «Коментарі», звідки він може переглядати коментарі, які були автоматично віднесені до певної групи. Доступні вкладки «Деактивовані» (рис. 5.6), звідки адміністратор може підтвердити або відновити автоматично класифікований коментар, «Заблоковані», звідки доступні коментарі, які адміністратор раніше підтвердив як ризиковані, «Відновлені», звідки доступні коментарі, які були після автоматичного віднесення до ризикованих, відновлені. Окрім того в меню доступні пункти: «Артисти», звідки можна керувати переліком артистів, які виступають на фестивалі, «Сцени», звідки можна керувати сценами фестивалю, «Тарифи», звідки можна керувати тарифами на продаж квитків на фестиваль, «Квитки», звідки можна керувати квитками, раніше придбаними користувачами, «Розклад», звідки можна керувати розкладом

виступів артистів, «Користувачі», звідки можна керувати доступом користувачів, «Форум», звідки можна керувати темами форуму, «Перевірка», звідки можна за потреби перевірити істинність квитка на фестиваль.

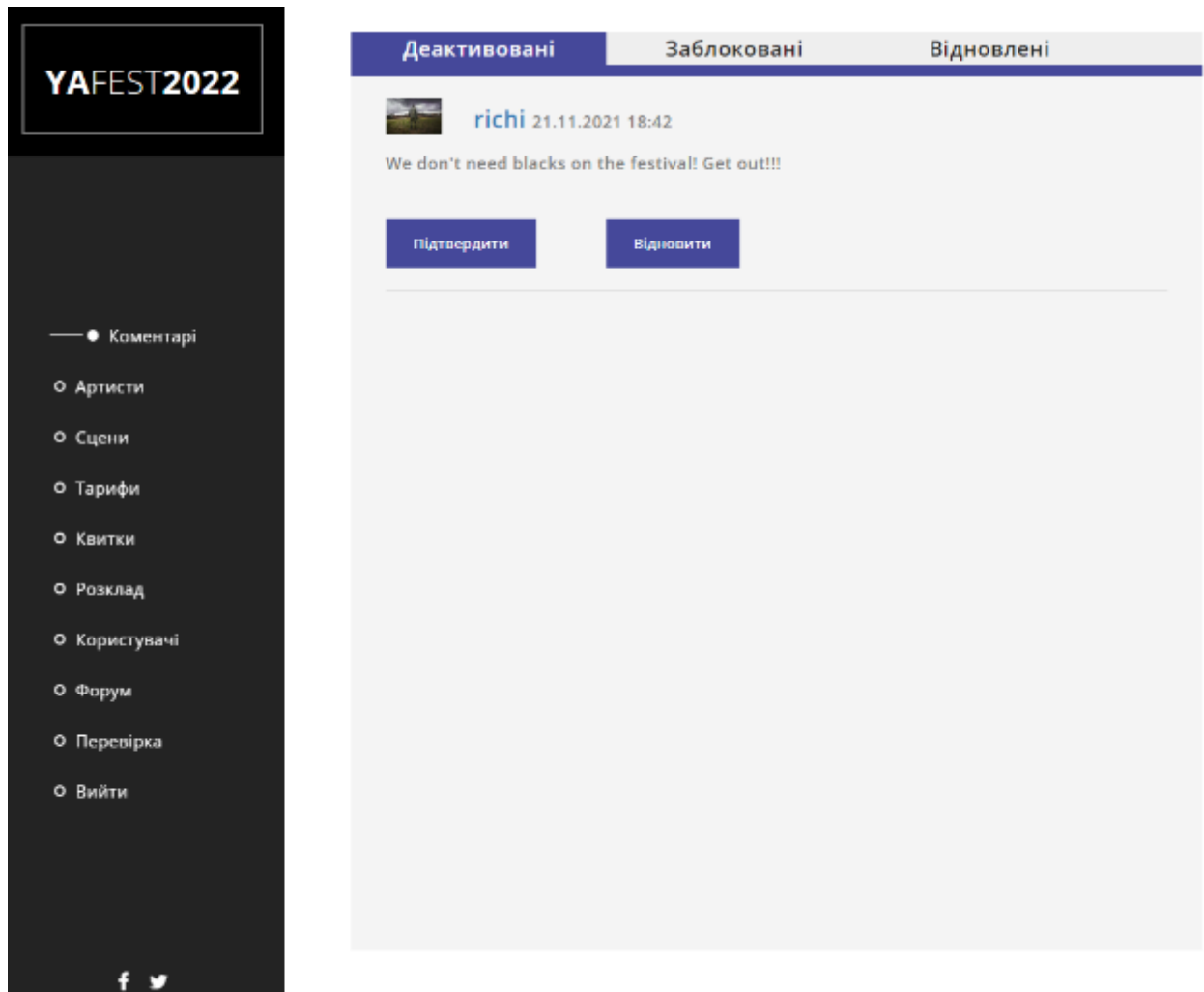


Рисунок 5.6 – Керування коментарями адміністратором

5.4 Експериментальне дослідження методу керування коментарями відвідувачів музичного фестивалю

Під час проведення експериментального дослідження розв’язувалась задача класифікації текстових повідомлень. Це дозволило порівнювати різні моделі класифікації (табл. 5.1), що використовуються в основі запропонованого методу.

Для проведення експериментального дослідження в основі було використано вибірки [20]-[21]. Для цього було об'єднано наявні повідомлення з внесенням додаткових, а далі вручну було визначено мітки для кожного повідомлення, що характеризують повідомлення як ризиковане або неризиковане. Ті повідомлення, які несуть ризик розповсюдження ворожнечі (а не просто злість), віднесені до класу ризикованих, усі інші – до іншого класу.

Таблиця 5.1 – Результати експериментального дослідження

Модель / метод	Точність, %
Наївний байєсів класифікатор	54,1
Запропонований метод керування коментарями відвідувачів музичного фестивалю	86,3

Отримані результати експериментального дослідження, по-перше, вказують на те, що використання Random Forest в основі запропонованого методу дозволило отримати результати, точність класифікації за якими значно переважає результати наївного байєсового класифікатора (майже в 1,6 разів), а по-друге, ці результати (86,3 %) є достатньо точними для практичного використання. Звичайно, що класифікація все ж може бути виконана невірно, для цього результати контролюються адміністратором, і стан, в якому відвідувач може бути позбавлений квитка, досягається тільки тоді, коли деяку кількість разів випадки ризикованих повідомлень були підтверджені адміністратором.

5.5 Висновки за розділом 5

Представлено результати розроблення програмного забезпечення на рівні взаємодії з користувачами різних типів у застосунку.

Виконано експериментальне дослідження класифікації повідомлень, за результатами якого виявилось, що запропонований метод керування коментарями відвідувачів музичного фестивалю дозволив отримати точність класифікації у 92,3 %.

ВИСНОВКИ

За результатами аналізу свободи вираження думок, дискримінації людей на основі мови ворожнечі, взаємовідношень між ними, а також огляду програмних засобів, які можуть використовуватися для підтримки відвідувачів музичного фестивалю, було обґрунтовано важливість інтеграції надання інформації про фестиваль, механізму продажу квитків та онлайнспілкування користувачів. Відповідно в розрізі даного програмного забезпечення важливою є наявність форуму та підтримка автоматизованого керування коментарями у ньому.

Для забезпечення керування коментарями відвідувачів музичного фестивалю було проаналізовано модель оброблення природної мови та моделі класифікації, в результаті чого було запропоновано власний метод. Наукова новизна роботи полягає в запропонованому методі керування коментарями відвідувачів музичного фестивалю, який ґрунтується на виконанні оброблення коментаря користувача моделлю GloVe з подальшою класифікацією методом Random Forest, що дозволяє визначати керуюче рішення з метою уникнення репутаційних втрат для фестивалю.

Для забезпечення створення на основі описаної концепції та запропонованого методу програмного забезпечення було виконано його проєктування. Для цього було обґрунтовано вибір мови програмування PHP для реалізації вебзастосунку, мови програмування Python для реалізації запропонованого методу, визначено функціональні вимоги до програмного забезпечення з моделюванням вимог на основі діаграм варіантів використання, виконано проєктування БД зі створенням її структури, визначено структуру програми. За результатами проєктування виконано програмну реалізацію зі створенням відповідного набору класів та визначенням станів коментаря відвідувача музичного фестивалю на основі діаграми станів.

Розроблене програмне забезпечення дозволяє переглядати інформацію про артистів, які беруть участь у фестивалі, розклад їх виступів, купувати квитки на фестиваль, створювати власний розклад відвідування фестивалю, переглядати і створювати теми у форумі, публікувати і переглядати коментарі в темах, оцінювати теми, уподобати коментарі. Експериментальне дослідження підтвердило, що запропонований метод дозволяє отримати придатну для використання на практиці точність класифікації коментарів.

Усі поставлені завдання на роботу виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Where do we draw the line between freedom of expression and discrimination? [Electronic resource]. – Access mode : <https://www.annefrank.org/en/topics/discrimination-and-equal-rights/where-do-we-draw-line-between-freedom-expression-and-discriminat/>.
2. Guo, L. Third-Person Effect and Hate Speech Censorship on Facebook [Electronic resource] / L. Guo, B.G. Johnson. 2020 // Social Media + Society. – Volume 6, Issue 2. – doi : <https://doi.org/10.1177/2056305120923003>.
3. Free speech & the law [Electronic resource] : “Hate Speech” & Non-Discrimination. – Access mode : <https://www.indexoncensorship.org/2020/01/free-speech-and-the-law-hate-speech-and-non-discrimination/>.
4. Mad Cool Festival [Electronic resource]. – Access mode : <https://madcoolfestival.es/en/index.php>.
5. Rock Werchter 2022 [Electronic resource]. – Access mode : <https://www.rockwerchter.be/en/>.
6. Primavera Sound [Electronic resource]. – Access mode : <https://www.primaverasound.com/en/>.
7. Venugopal, K. Mathematical Introduction to GloVe Word Embedding [Electronic resource] / K. Venugopal. – Access mode : <https://becominghuman.ai/mathematical-introduction-to-glove-word-embedding-60f24154e54c>.
8. Ganegedara, T. Intuitive Guide to Understanding GloVe Embeddings [Electronic resource] / T. Ganegedara. – Access mode : <https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010>.
9. Pennington, J. GloVe [Text] : Global Vectors for Word Representation / Jeffrey Pennington, Richard Socher, Christopher D. Manning // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), October 25-29, 2014, Doha, Qatar. – Doha : SIGDAT,

2014. – Pp. 1532–1543.

10. Donges, N. A Complete Guide to the Random Forest Algorithm [Electronic resource] / N. Donges. – Access mode : <https://builtin.com/data-science/random-forest-algorithm>.
11. Introduction to Random Forest in Machine Learning [Electronic resource]. – Access mode : <https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/>.
12. Breiman, L. Random Forests [Electronic resource] / Leo Breiman and Adele Cutler. – Access mode : https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.
13. What is Random Forest? [Electronic resource]. – Access mode : <https://www.ibm.com/cloud/learn/random-forest>.
14. Stellman, A. Head First C# [Text] : A Learner's Guide to Real-World Programming with C# and .NET Core / Andrew Stellman, Jennifer Greene. – Sebastopol, CA : O'Reilly Media, 2021. – 800 p.
15. Nixon, R. Learning PHP, MySQL & JavaScript [Text] : With jQuery, CSS & HTML5 / R. Nixon. – Sebastopol : O'Reilly, 2018. – 832 p.
16. Кузнецов, М. Самоучитель PHP 7 [Текст] / М. Кузнецов, И. Симдянов. – СПб. : БХВ-Петербург, 2018. – 448 с.
17. Lubanovic, B. Introducing Python : Modern Computing in Simple Packages [Text] / B. Lubanovic. – Sebastopol, CA : O'Reilly Media, 2019. – 630 p.
18. Lutz, M. Programming Python [Text] : Powerful Object-Oriented Programming / M. Lutz. – Sebastopol, CA : O'Reilly Media, 2011. – 1632 p.
19. Free 518 Template Sentra [Electronic resource]. – Access mode : <https://templatemo.com/tm-518-sentra>.
20. Samoshyn, A. Hate Speech and Offensive Language Dataset [Electronic resource] / A. Samoshyn. – Access mode : <https://www.kaggle.com/mrmorj/hate-speech-and-offensive-language-dataset>.
21. Emotions in text [Electronic resource]. – Access mode : <https://www.kaggle.com/ishantjuyal/emotions-in-text>.

ДОДАТОК А
Технічне завдання

Вступ

Сучасний музичний фестиваль потребує відповідного програмного забезпечення для підтримки взаємодії з користувачем щодо його інформування про події фестивалю, придбання квитків, надання майданчика для взаємодії з іншими користувачами.

Одночасно такий майданчик не може бути залишений без керування, адже спілкування відвідувачів без будь-якого відслідковування може нести репутаційні ризики для фестивалю, що може призвести до скасування його проведення. Відповідно підтримку цієї діяльності має забезпечувати програмне забезпечення, що розробляється.

A.1 Підстави для розробки

Підставою для розробки є завдання на кваліфікаційну роботу магістра, визначене за темою «Дослідження та програмна реалізація методів керування коментарями відвідувачів музичного фестивалю», яку затверджено наказом № 435 від 5 листопада 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програмна розробка призначена для:

- надання користувачам, які є потенційними відвідувачами фестивалю, інформації про музичний фестиваль;
- створення майданчика для комунікації відвідувачів фестивалю з можливістю керування ризикованим спілкуванням, отримання власних даних, пов'язаних з фестивалем.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати такі функціональні характеристики:

- створення тем для спілкування на форумі фестивалю;
- публікація коментарів в темі форуму;
- уподобання коментарів;
- невподобання коментарів;
- відображення коментарів у темі форуму;
- відображення тем форуму;
- голосування за тему форуму;
- відслідковування переглядів теми форуму;
- пошук тем форуму;
- відображення розкладу виступу артистів;
- придбання квитків;
- заборона на доступ користувачу;
- виявлення повідомлень з репутаційним ризиком;
- визначення і відображення артистів;
- визначення часу і місця виступу артистів;
- визначення сцен фестивалю;
- скасування квитка;
- створення і відображення персонального розкладу відвідування фестивалю;
- уподобання виступу артиста;
- відображення власних голосів за теми форуму;
- відображення оцінок власних коментарів;
- відображення оцінених користувачем коментарів;
- відображення квитків користувача;
- визначення груп тарифів квитків;
- визначення вартісних пропозицій для груп квитків;

- відображення варіантів придбання квитків;
- відображення і керування заблокованими користувачами;
- відображення і керування скасованими квитками;
- відображення і керування повідомленнями з репутаційним ризиком;
- авторизація і реєстрація користувачів.

A.3.2 Вимоги до складу та параметрів технічних засобів

Апаратне забезпечення роботи вебсервера повинно відповідати вимогам роботи з щонайменше 200 користувачами одночасно. Для підтримки цих вимог апаратне забезпечення повинно мінімально включати:

- процесор: чотири ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять: 10 Гб;
- жорсткий диск: 300 Мб доступного простору.

A.4 Порядок контролю та приймання

Календарний план є основою для контролю виконання роботи, у свою чергу приймання роботи здійснюється під час її захисту перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

Б.1 Текст файла ArtistModel.php

```

<?php

class ArtistModel {

    public static function add ($name, $country, $level,
    $description, $site, $ymsite, $spotsite, $fbpage, $inpage, $youpage,
    $youvideo) {

        $dbc = DBMFest::getConnection();

        $sql = "
            INSERT INTO Artist(name, country, level,
description, site, ymsite, spotsite, fbpage, inpage, youpage, youvideo)
            VALUES (:name, :country, :level, :description,
:site, :ymsite, :spotsite, :fbpage, :inpage, :youpage, :youvideo)
            ";

        $result = $dbc->prepare($sql);

        $result->bindParam(':name', $name, PDO::PARAM_STR);
        $result->bindParam(':level', $level, PDO::PARAM_INT);
        $result->bindParam(':country', $country,
PDO::PARAM_STR);
        $result->bindParam(':description', $description,
PDO::PARAM_STR);
        $result->bindParam(':site', $site, PDO::PARAM_STR);
        $result->bindParam(':ymsite', $ymsite, PDO::PARAM_STR);
        $result->bindParam(':spotsite', $spotsite,
PDO::PARAM_STR);
        $result->bindParam(':fbpage', $fbpage, PDO::PARAM_STR);
        $result->bindParam(':inpage', $inpage, PDO::PARAM_STR);
        $result->bindParam(':youpage', $youpage,
PDO::PARAM_STR);
        $result->bindParam(':youvideo', $youvideo,
PDO::PARAM_STR);

        return $result->execute();

    }

    public static function edit ($id, $name, $country, $level,
    $description, $site, $ymsite, $spotsite, $fbpage, $inpage, $youpage,
    $youvideo) {

```

```

$dbc = DBMFest::getConnection();

$sql = "
    UPDATE Artist
    SET
        name = :name,
        country = :country,
        level = :level,
        description = :description,
        site = :site,
        ymsite = :ymsite,
        spotsite = :spotsite,
        fbpage = :fbpage,
        inpage = :inpage,
        youpage = :youpage,
        youvideo = :youvideo
    WHERE id = :id
    ";

$result = $dbc->prepare($sql);

$result->bindParam(':id', $id, PDO::PARAM_INT);
$result->bindParam(':name', $name, PDO::PARAM_STR);
$result->bindParam(':level', $level, PDO::PARAM_INT);
$result->bindParam(':country', $country, PDO::PARAM_STR);
$result->bindParam(':description', $description,
PDO::PARAM_STR);
$result->bindParam(':site', $site, PDO::PARAM_STR);
$result->bindParam(':ymsite', $ymsite, PDO::PARAM_STR);
$result->bindParam(':spotsite', $spotsite, PDO::PARAM_STR);
$result->bindParam(':fbpage', $fbpage, PDO::PARAM_STR);
$result->bindParam(':inpage', $inpage, PDO::PARAM_STR);
$result->bindParam(':youpage', $youpage, PDO::PARAM_STR);
$result->bindParam(':youvideo', $youvideo, PDO::PARAM_STR);

return $result->execute();
}

public static function getInformation ($id) {

    $dbc = DBMFest::getConnection();

    $sql = "
        SELECT * FROM Artist WHERE id = ':id'
        ";

    $result = $dbc->prepare($sql);

```

```

$result->bindParam(':id', $id, PDO::PARAM_INT);

$result->execute();

return $result->fetch(PDO::FETCH_ASSOC);    }

}

```

```

public static function getArtists () {

    $dbc = DBMFest::getConnection();

    $sql = "
        SELECT id, name, country, description, youvideo,
site
            FROM Artist
            ORDER BY name
        ";

    $artists = array();

    $result = $dbc->query($sql);

    $artists = $result->fetchAll(PDO::FETCH_ASSOC);

    return $artists;
}

```

```

public static function getArtistsLevel ($level) {

    $dbc = DBMFest::getConnection();

    $sql = "
        SELECT id, name, country, description, youvideo,
site
            FROM Artist
            WHERE level = $level
            ORDER BY name
        ";

    $artists = array();

    $result = $dbc->query($sql);

    $artists = $result->fetchAll(PDO::FETCH_ASSOC);

    return $artists;
}

```

```

    }

    public static function searchArtist ($name) {

        $dbc = DBMFest::getConnection();

        $sql = "
            SELECT id, name, country, description, youvideo,
site
            FROM Artist
            WHERE name LIKE '%$name%'
            ORDER BY name
            ";

        $artists = array();

        $result = $dbc->query($sql);

        $artists = $result->fetchAll(PDO::FETCH_ASSOC);

        return $artists;
    }

}

```

Б.2 Текст файла TopicModel.php

```

<?php

class TopicModel {

    public static function addTopic ($name, $userId) {

        $dbc = DBMFest::getConnection();

        $sql = $dbc->prepare("SELECT User.name FROM User INNER JOIN
Ticket on Ticket.userId = User.id WHERE User.id = :us AND Ticket.state =
1");

        $sql->bindParam(':us',$userId, PDO::PARAM_INT);

        $sql->execute();

        if ($sql->rowCount()>0) {
            $today = date('Y-m-d H:i:s');

            $sql = "

```

```

        INSERT INTO Topic(name, userId, votes, views, when)
        VALUES (:topicname, :user, :qvotes, :qviews, :date)
        ";

        $result = $dbc->prepare($sql);

        $result->bindParam(':topicname', $name,
PDO::PARAM_STR);
        $result->bindParam(':user', $userId, PDO::PARAM_INT);
        $result->bindParam(':qvotes', 0, PDO::PARAM_INT);
        $result->bindParam(':qviews', 0, PDO::PARAM_INT);
        $result->bindParam(':date', $today, PDO::PARAM_STR);

        return $result->execute();
    }
}

public static function canAddTopic ($userId) {
    $dbc = DBMFest::getConnection();

    $sql = $dbc->prepare("SELECT User.name FROM User INNER JOIN
Ticket on Ticket.userId = User.id WHERE User.id = :us AND Ticket.state =
1");

    $sql->bindParam(':us', $userId, PDO::PARAM_INT);

    $sql->execute();

    if ($sql->rowCount() > 0) {
        return true;
    }
    else {
        return false;
    }
}

public static function getTopicComments ($topic) {

    $dbc = DBMFest::getConnection();

    $sql = "
        SELECT Comment.id, Comment.userId, Comment.topicId,
        DATE_FORMAT(`Comment.when`, '%d.%m.%Y %H:%i') AS
when_format,
        Comment.likes, Comment.dislikes, Comment.text,
        Comment.state, User.login, User.photo
        FROM Comment
        INNER JOIN User on Comment.userId = User.id
        WHERE topicId = $topic AND (Comment.state = 1 OR
Comment.state = 4)
    ";

```

```

        ORDER BY Comment.when_format
    ";

    $comments = array();

    $result = $dbc->query($sql);

    $comments = $result->fetchAll(PDO::FETCH_ASSOC);

    return $comments;
}

public static function getOwnComments ($id) {

    $dbc = DBMFest::getConnection();

    $sql = "
        SELECT Comment.id, Comment.userId, Comment.topicId,
        DATE_FORMAT(`Comment.when`, '%d.%m.%Y %H:%i') AS
when_format,

        Comment.likes, Comment.dislikes, Comment.text,
        Comment.state
        FROM Comment
        INNER JOIN Topic on Comment.userId = User.id
        WHERE userId = $id
        ORDER BY Comment.when_format DESC
    ";

    $comments = array();

    $result = $dbc->query($sql);

    $comments = $result->fetchAll(PDO::FETCH_ASSOC);

    return $comments;
}

public static function getCommentState ($state) {
    switch ($state) {
        case '1' :
            return 'активне';
            break;
        case '2' :
            return 'заблоковане';
            break;
        case '3' :
            return 'деактивоване';
            break;
        case '4' :

```

```

        return 'Відновлене';
        break;
    }
}

public static function getTopicsByDate ($up) {

    $dbc = DBMFest::getConnection();

    $sql = "
        SELECT Topic.id, Topic.name, Topic.userId,
            Topic.votes, Topic.views,
            DATE_FORMAT(`Topic.when`, '%d.%m.%Y %H:%i') AS
when_format,

            User.login, User.photo
        FROM Topic
        INNER JOIN User on Topic.userId = User.id
        ORDER BY Topic.when_format DESC
    ";

    if ($up) {
        $sql = "
            SELECT Topic.id, Topic.name, Topic.userId,
                Topic.votes, Topic.views,
                DATE_FORMAT(`Topic.when`, '%d.%m.%Y %H:%i') AS
when_format,

                User.login, User.photo
            FROM Topic
            INNER JOIN User on Topic.userId = User.id
            ORDER BY Topic.when_format
        ";
    }
    $topics = array();

    $result = $dbc->query($sql);

    $topics = $result->fetchAll(PDO::FETCH_ASSOC);

    return $topics;
}

public static function getTopicsByVotes ($up) {

    $dbc = DBMFest::getConnection();

    $sql = "
        SELECT Topic.id, Topic.name, Topic.userId,
            Topic.votes, Topic.views,

```

```

                                DATE_FORMAT(`Topic.when`, '%d.%m.%Y %H:%i') AS
when_format,
                                User.login, User.photo
                                FROM Topic
                                INNER JOIN User on Topic.userId = User.id
                                ORDER BY Topic.votes DESC
                                ";

if ($up) {
    $sql = "
        SELECT Topic.id, Topic.name, Topic.userId,
        Topic.votes, Topic.views,
        DATE_FORMAT(`Topic.when`, '%d.%m.%Y %H:%i') AS
when_format,
        User.login, User.photo
        FROM Topic
        INNER JOIN User on Topic.userId = User.id
        ORDER BY Topic.votes
        ";
    }

    $topics = array();

    $result = $dbc->query($sql);

    $topics = $result->fetchAll(PDO::FETCH_ASSOC);

    return $topics;
}

public static function editTopic ($id, $tname){

    $dbc = DBMFest::getConnection();

    $sql = "
        UPDATE Topic
        SET
            name = :name
        WHERE id = :id
        ";

    $result = $dbc->prepare($sql);

    $result->bindParam(':id', $id, PDO::PARAM_INT);
    $result->bindParam(':name', $tname, PDO::PARAM_STR);

    return $result->execute();
}

```

```

public static function getTopicsByViews ($up) {

    $dbc = DBMFest::getConnection();

    $sql = "
        SELECT Topic.id, Topic.name, Topic.userId,
            Topic.votes, Topic.views,
            DATE_FORMAT(`Topic.when`, '%d.%m.%Y %H:%i') AS
when_format,

            User.login, User.photo
        FROM Topic
        INNER JOIN User on Topic.userId = User.id
        ORDER BY Topic.views DESC
    ";

    if ($up) {
        $sql = "
            SELECT Topic.id, Topic.name, Topic.userId,
                Topic.votes, Topic.views,
                DATE_FORMAT(`Topic.when`, '%d.%m.%Y %H:%i') AS
when_format,

                User.login, User.photo
            FROM Topic
            INNER JOIN User on Topic.userId = User.id
            ORDER BY Topic.views
        ";
    }

    $topics = array();

    $result = $dbc->query($sql);

    $topics = $result->fetchAll(PDO::FETCH_ASSOC);

    return $topics;
}

public static function searchTopic ($title) {

    $dbc = DBMFest::getConnection();

    $sql = "
        SELECT Topic.id, Topic.name, Topic.userId,
            Topic.votes, Topic.views,
            DATE_FORMAT(`Topic.when`, '%d.%m.%Y %H:%i') AS
when_format,

            User.login, User.photo
        FROM Topic
        INNER JOIN User on Topic.userId = User.id
    
```

```

        WHERE Topic.name LIKE '%$title%'
        ORDER BY Topic.id DESC
    ";

    $topics = array();

    $result = $dbc->query($sql);

    $topics = $result->fetchAll(PDO::FETCH_ASSOC);

    return $topics;
}

public static function voteTopic ($us, $topic){

    $today = date('Y-m-d H:i:s');

    $dbc = DBMFest::getConnection();

    $sql = "
        INSERT INTO Vote(userId, topicId, when)
        VALUES (:us, :, :topic, :today)
    ";

    $result = $dbc->prepare($sql);
    $result->bindParam(':us', $us, PDO::PARAM_INT);
    $result->bindParam(':topic', $topic, PDO::PARAM_INT);
    $result->bindParam(':today', $today, PDO::PARAM_STR);

    $res = $result->execute();

    $sql = "
        UPDATE Topic
        SET
            votes = votes + 1
        WHERE id = :topic
    ";

    $result = $dbc->prepare($sql);

    $result->bindParam(':topic', $topic, PDO::PARAM_INT);

    $result->execute();

    return $res;
}

public static function viewTopic ($us, $topic){

```

```

$today = date('Y-m-d H:i:s');

$dbc = DBMFest::getConnection();

$sql = "
    INSERT INTO View(userId, topicId, when)
    VALUES (:us, :, :topic, :today)
    ";

$result = $dbc->prepare($sql);
$result->bindParam(':us', $us, PDO::PARAM_INT);
$result->bindParam(':topic', $topic, PDO::PARAM_INT);
$result->bindParam(':today', $today, PDO::PARAM_STR);

$res = $result->execute();

$sql = "
    UPDATE Topic
    SET
        views = views + 1
    WHERE id = :topic
    ";

$result = $dbc->prepare($sql);

$result->bindParam(':topic', $topic, PDO::PARAM_INT);

$result->execute();

return $res;
}

public static function getOwnTopics ($id) {

    $dbc = DBMFest::getConnection();

    $sql = "
        SELECT id, name, userId, votes, views,
        DATE_FORMAT(`when`, '%d.%m.%Y %H:%i') AS
when_format,

        FROM Topic
        WHERE userId = $id
        ORDER BY id DESC
    ";

    $topics = array();

    $result = $dbc->query($sql);

```

```

        $topics = $result->fetchAll(PDO::FETCH_ASSOC);

        return $topics;
    }

    public static function getOwnVotes ($us) {

        $dbc = DBMFest::getConnection();

        $sql = "
            SELECT Vote.id, Vote.userId, Vote.topicId,
            DATE_FORMAT(`Vote.when`, '%d.%m.%Y %H:%i') AS
when_format,
            Topic.id, Topic.name, Topic.userId, Topic.votes,
            Topic.views,
            DATE_FORMAT(`Topic.when`, '%d.%m.%Y %H:%i') AS
twhen_format
            FROM Vote
            INNER JOIN Topic on Topic.userId = Vote.userId
            WHERE userId = $us
            ORDER BY Vote.id DESC
        ";

        $topics = array();

        $result = $dbc->query($sql);

        $topics = $result->fetchAll(PDO::FETCH_ASSOC);

        return $topics;
    }

    public static function unVoteTopic ($us, $topic){

        $dbc = DBMFest::getConnection();

        $sql = "
            DELETE FROM Vote WHERE Vote.userId = :us AND
Vote.topicId = :topic
        ";

        $result = $dbc->prepare($sql);
        $result->bindParam(':us', $us, PDO::PARAM_INT);
        $result->bindParam(':topic', $topic, PDO::PARAM_INT);

        $res = $result->execute();

        $sql = "
            UPDATE Topic

```

```

        SET
            votes = votes - 1
        WHERE id = :topic
    ";

    $result = $dbc->prepare($sql);

    $result->bindParam(':topic', $topic, PDO::PARAM_INT);

    $result->execute();

    return $res;
}

public static function likeComment ($us, $comm, $type){

    $today = date('Y-m-d H:i:s');

    if ($type) {
        $typeint = 1;
    }
    else {
        $typeint = 0;
    }

    $dbc = DBMFest::getConnection();

    $likefl = 0
    $sql = $dbc->prepare("SELECT Liker.id FROM Liker WHERE
userId = :us AND commentId = :comm AND type = 1");
    $sql->bindParam(':us',$userId, PDO::PARAM_INT);
    $sql->bindParam(':comm', $comm, PDO::PARAM_INT);

    $sql->execute();

    if ($sql->rowCount()>0) {
        $likefl = 1;
    }
    else {
        $sql = $dbc->prepare("SELECT Liker.id FROM Liker WHERE
userId = :us AND commentId = :comm AND type = 0");
        $sql->bindParam(':us',$userId, PDO::PARAM_INT);
        $sql->bindParam(':comm', $comm, PDO::PARAM_INT);

        $sql->execute();

        if ($sql->rowCount()>0) {
            $likefl = 2;
        }
    }
}

```

```

    }

    if ($likefl == 0) {
        $sql = "
            INSERT INTO Liker(userId, commentId, when, type)
            VALUES (:us, :, :comm, :today, :type)
            ";

        $result = $dbc->prepare($sql);
        $result->bindParam(':us', $us, PDO::PARAM_INT);
        $result->bindParam(':comm', $comm, PDO::PARAM_INT);
        $result->bindParam(':today', $today, PDO::PARAM_STR);
        $result->bindParam(':type', $typeint, PDO::PARAM_INT);

        $res = $result->execute();

        $sql = "
            UPDATE Comment
            SET
                likes = likes + 1
            WHERE id = :comm
            ";

        if ($typeint == 0) {
            $sql = "
                UPDATE Comment
                SET
                    dislikes = dislikes + 1
                WHERE id = :comm
                ";
        }

        $result = $dbc->prepare($sql);

        $result->bindParam(':comm', $comm, PDO::PARAM_INT);

        $result->execute();

        return $res;
    }
    else if (($likefl == 2 && $typeint == 1) || ($likefl == 1
&& $typeint == 0)) {
        $sql = "
            UPDATE Liker
            SET type = :type
            WHERE commentId = :comm AND userId = :us
            ";
        $sql->bindParam(':us', $userId, PDO::PARAM_INT);
        $sql->bindParam(':comm', $comm, PDO::PARAM_INT);
    }
}

```

```

$sql->bindParam(':type', $typeint, PDO::PARAM_INT);

$res = $sql->execute();

$sql = "
    UPDATE Comment
    SET
        likes = likes + 1,
        dislikes = dislikes - 1
    WHERE id = :comm
";

if ($typeint == 0) {
    $sql = "
        UPDATE Comment
        SET
            dislikes = dislikes + 1,
            likes = likes - 1
        WHERE id = :comm
    ";
}

$result = $dbc->prepare($sql);

$result->bindParam(':comm', $comm, PDO::PARAM_INT);

$result->execute();

return $res;
}
else {
    $sql = "
        UPDATE Comment
        SET
            likes = likes - 1
        WHERE id = :comm
    ";

    if ($typeint == 0) {
        $sql = "
            UPDATE Comment
            SET
                dislikes = dislikes - 1
            WHERE id = :comm
        ";
    }

    $result = $dbc->prepare($sql);

```

```

        $result->bindParam(':comm', $comm,
PDO::PARAM_INT);

        return $result->execute();

    }
}

public static function editComment ($id, $text){

    $dbc = DBMFest::getConnection();

    $sql = "
        UPDATE Comment
        SET
            text = :text
        WHERE id = :id
    ";

    $result = $dbc->prepare($sql);

    $result->bindParam(':id', $id, PDO::PARAM_INT);
    $result->bindParam(':text', $text, PDO::PARAM_STR);

    return $result->execute();
}

public static function getLikedComments ($us, $type) {

    $dbc = DBMFest::getConnection();

    $sql = "
        SELECT Liker.commentId, Liker.Id, Liker.userId,
        Liker.type, DATE_FORMAT(`Liker.when`, '%d.%m.%Y
%H:%i') AS when_format,
        Comment.Id, Comment.text, Comment.topicId
        FROM Liker
        INNER JOIN Comment on Comment.id = Liker.commentId
        WHERE Liker.userId = $id AND Liker.type = $type
        ORDER BY Comment.when_format DESC
    ";

    $comments = array();

    $result = $dbc->query($sql);

    $comments = $result->fetchAll(PDO::FETCH_ASSOC);

    return $comments;
}

```

```

    }

    public static function getRiskedComments ($state) {

        $statecode = 3;

        if ($state == 1) {
            $statecode = 4;
        }
        else if ($state == 2) {
            $statecode = 2;
        }

        $dbc = DBMFest::getConnection();

        $sql = "
            SELECT Comment.id, Comment.userId, Comment.topicId,
            DATE_FORMAT(`Comment.when`, '%d.%m.%Y %H:%i') AS
when_format,
            Comment.likes, Comment.dislikes, Comment.text,
            Comment.state, User.login, User.photo
            FROM Comment
            INNER JOIN User on Comment.userId = User.id
            WHERE state = $statecode
            ORDER BY Comment.when_format DESC
            ";

        $comments = array();

        $result = $dbc->query($sql);

        $comments = $result->fetchAll(PDO::FETCH_ASSOC);

        return $comments;
    }

    public static function addComment ($text, $userId, $topicId) {

        $dbc = DBMFest::getConnection();

        $today = date('Y-m-d H:i:s');

        $sql = $dbc->prepare("SELECT id, userId, from, to, state
            FROM Ban
            WHERE userId = :userId AND state = 1
            AND from > :today AND to < :today");
        $sql->bindParam(':us',$userId, PDO::PARAM_INT);
        $sql->bindParam(':today',$today, PDO::PARAM_STR);
    }

```

```

$sql->execute();

if ($sql->rowCount() == 0) {

    $state = CommentManager::getDecision($text);

    $sql = "
        INSERT INTO Comment(userId, topicId, likes,
dislikes, text, state, when)
        VALUES (:userId, :topicId, 0, 0, :text, :state,
:today)
        ";

    $result = $dbc->prepare($sql);

    $result->bindParam(':userId', $userId, PDO::PARAM_INT);
    $result->bindParam(':topicId', $topicId,
PDO::PARAM_INT);
    $result->bindParam(':text', $text, PDO::PARAM_STR);
    $result->bindParam(':state', $state, PDO::PARAM_INT);
    $result->bindParam(':when', $today, PDO::PARAM_STR);

    if ($state != 1) {
        $sql = "
            UPDATE User
            SET
                rating = rating - 1
            WHERE id = :id
            ";

        $res = $dbc->prepare($sql);

        $res->bindParam(':id', $userId, PDO::PARAM_INT);

        $res->execute();
    }

    return $result->execute();
}

}

public static function editUserRating ($us, $rating){

    $dbc = DBMFest::getConnection();

    $sql = "
        UPDATE User
        SET
            rating = :rating
    
```

```

        WHERE id = :us
        ";

$result = $dbc->prepare($sql);

$result->bindParam(':us', $us, PDO::PARAM_INT);
$result->bindParam(':rating', $rating, PDO::PARAM_INT);

$result->execute();

return $res;
}

public static function cancelBan ($us){

    $today = date('Y-m-d H:i:s');

    $dbc = DBMFest::getConnection();

    $sql = "
        UPDATE Ban
        SET
            state = 0
        WHERE userId = :us AND from > :today AND to <
:today
        ";

    $result = $dbc->prepare($sql);

    $result->bindParam(':us', $us, PDO::PARAM_INT);
    $result->bindParam(':today', $today, PDO::PARAM_STR);

    $result->execute();

    return $res;
}

public static function banUser ($us, $from, $to) {

    $dbc = DBMFest::getConnection();

    $sql = "
        INSERT INTO Ban(userId, from, to, state)
        VALUES (:userId, :from, :to, 1)
        ";

    $result = $dbc->prepare($sql);

    $result->bindParam(':userId', $us, PDO::PARAM_INT);

```

```

        $result->bindParam(':from', $from, PDO::PARAM_STR);
        $result->bindParam(':to', $to, PDO::PARAM_STR);

        return $result->execute();
    }
}

public static function approveComment ($id, $state){

    $stateint = 2;
    if (!$state) {
        $stateint = 4;
    }
    $dbc = DBMFest::getConnection();

    $sql = "
        UPDATE Comment
        SET
            state = :stateint
        WHERE id = :id
    ";

    $result = $dbc->prepare($sql);

    $result->bindParam(':id', $id, PDO::PARAM_INT);
    $result->bindParam(':stateint', $stateint, PDO::PARAM_INT);

    return $result->execute();
}
}

```

ДОДАТОК В
Слайди презентації

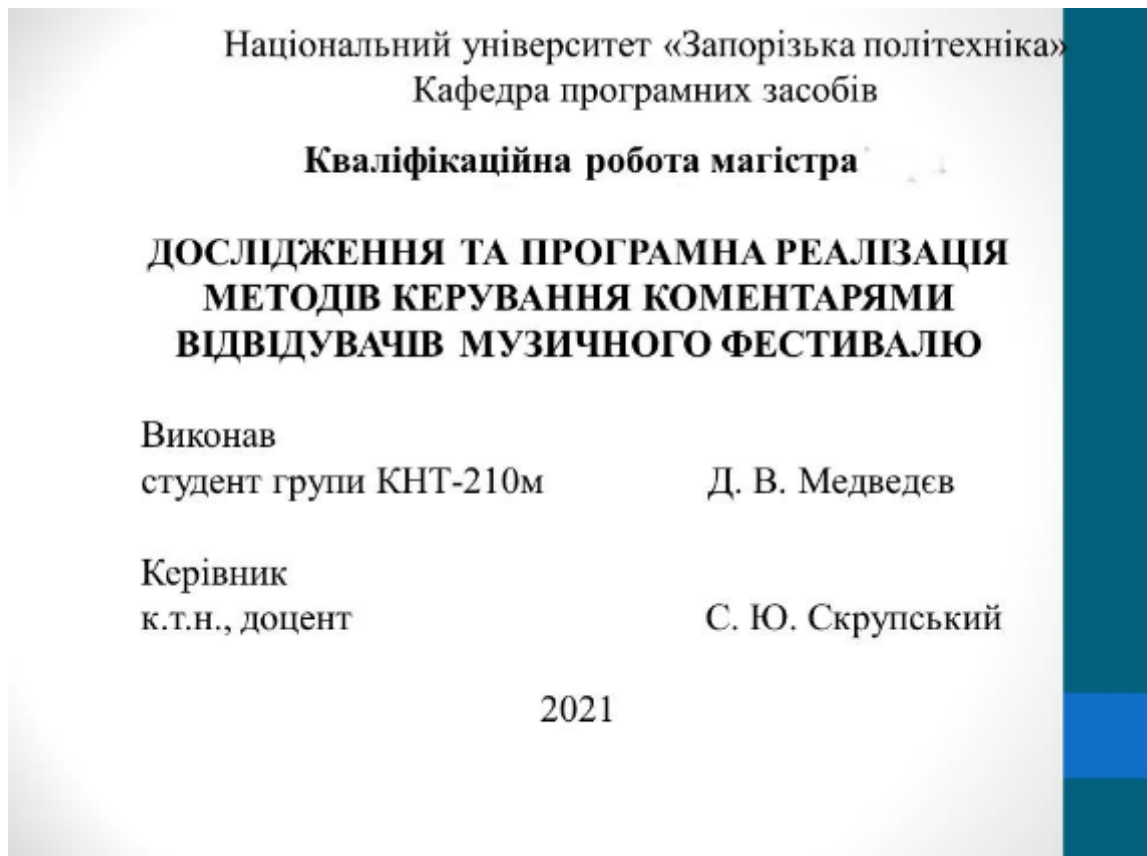


Рисунок В.1 – Слайд 1



Рисунок В.2 – Слайд 2

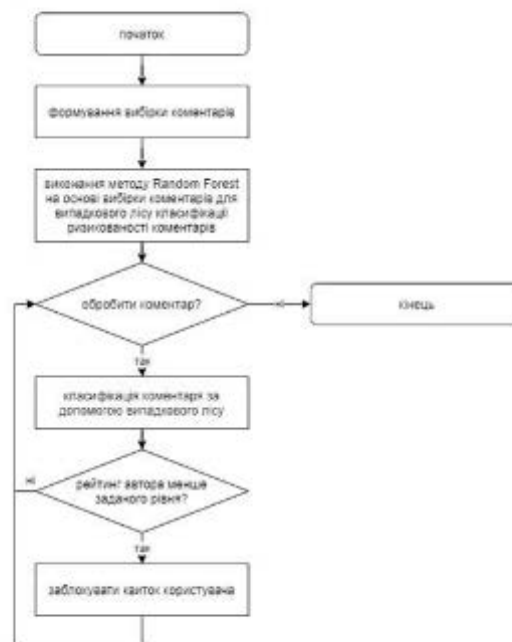
Завдання роботи

- ✓ провести дослідження методів керування коментарями відвідувачів музичного фестивалю;
- ✓ виконати проєктування програмного забезпечення;
- ✓ реалізувати програмне забезпечення.

[3]

Рисунок В.3 – Слайд 3

Метод керування коментарями відвідувачів музичного фестивалю



[4]

Рисунок В.4 – Слайд 4

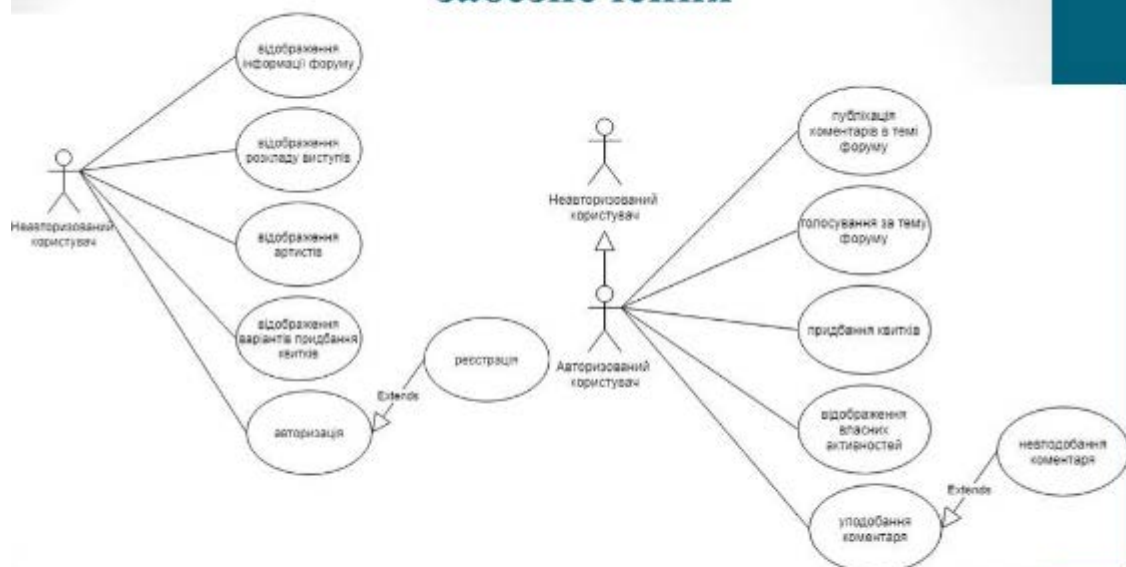
Вибір мови програмування

Критерій	C#	PHP
Об'єктно-орієнтована парадигма програмування	Так	Так
Кількість розробників	Менша	Більша
Відкритість програмного забезпечення	Ні	Так
Незалежність від платформи	Неповна, розвивається	Так
Хостинг	Більш коштовний	Менш коштовний
Підтримка веброзробки	Так	Так

5

Рисунок В.5 – Слайд 5

Функціональні вимоги до програмного забезпечення



6

Рисунок В.6 – Слайд 6

Функціональні вимоги до програмного забезпечення

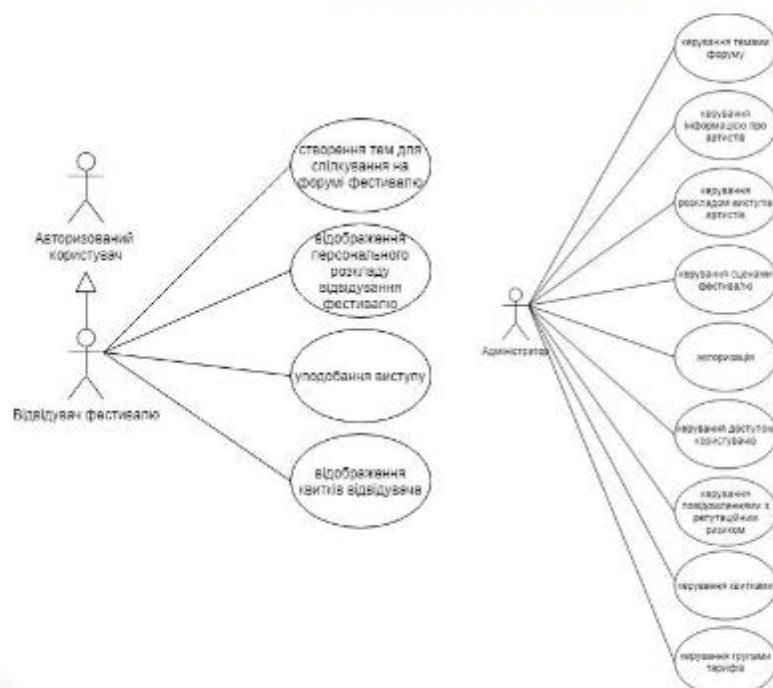


Рисунок В.7 – Слайд 7

Структурна схема бази даних

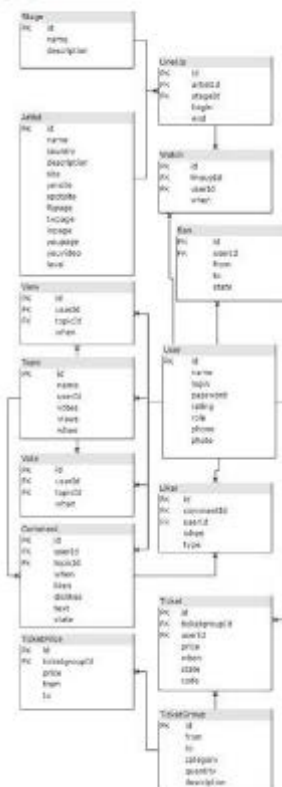


Рисунок В.8 – Слайд 8

```

graph TD
    systemapp --> Model
    systemapp --> Controller
    systemapp --> indexphp[index.php]
    systemapp --> Views

    Model --> ArtistModelphp[ArtistModel.php]
    Model --> TopicModelphp[TopicModel.php]
    Model --> LineupModelphp[LineupModel.php]
    Model --> Topicdetailphp[Topicdetail.php]
    Model --> ImageModelphp[ImageModel.php]
    Model --> UserModelphp[UserModel.php]

    Controller --> UserControllerphp[UserController.php]
    Controller --> VisitorControllerphp[VisitorController.php]
    Controller --> AdminControllerphp[AdminController.php]
    Controller --> SignInControllerphp[SignInController.php]
    Controller --> LoginControllerphp[LoginController.php]
    Controller --> LogoutControllerphp[LogoutController.php]

    indexphp --> Components
    indexphp --> Template

    Components --> DBMFeedphp[DBMFeed.php]
    Components --> CommentFormMethodphp[CommentFormMethod.php]

    Template --> css
    Template --> image
    Template --> js

    Views --> artistphp1[artist.php]
    Views --> artistphp2[artist.php]
    Views --> topicsphp[topics.php]
    Views --> topicdetailphp[topicdetail.php]
    Views --> lineupphp[lineup.php]
    Views --> user_lineupphp[user_lineup.php]
    Views --> forumtopicsphp[forumtopics.php]
    Views --> forumcommentsphp[forumcomments.php]
    Views --> loginphp[login.php]
    Views --> userindexphp[userindex.php]
    Views --> visitortopicphp[visitortopic.php]
    Views --> usercommentsphp[usercomments.php]
    Views --> xartistphp[xartist.php]
    Views --> xarttopicsphp[xarttopics.php]
    Views --> userdetailsphp[userdetails.php]
    Views --> adminnewartistphp[adminnewartist.php]
    Views --> adminindexlistphp[adminindexlist.php]
    Views --> adminnocommentsphp[adminnocomments.php]
    Views --> adminusersphp[adminusers.php]
    Views --> adminreportsphp[adminreports.php]
    Views --> adminindexphp[adminindex.php]
    Views --> adminlineupphp[adminlineup.php]
  
```

10

```
graph TD; Start(( )) --> Step1[Користуван публікує коментар]; Step1 --> Step2[Незафіксований]; Step2 -->|Аналіз коментаря запропонованим методом не впливає ризиків| Step3[Активний]; Step2 -->|Запропонований метод розцінює коментар як потенційно ризикований| Step4[Деактивований]; Step4 -->|Адміністратор підтверджує ризикованість коментаря| Step5[Заблокований]; Step4 -->|Адміністратор оцінює ризикованість коментаря як недостатньо високу| Step6[Відновлений]; Step3 --> End(( )); Step5 --> End; Step6 --> End;
```

10

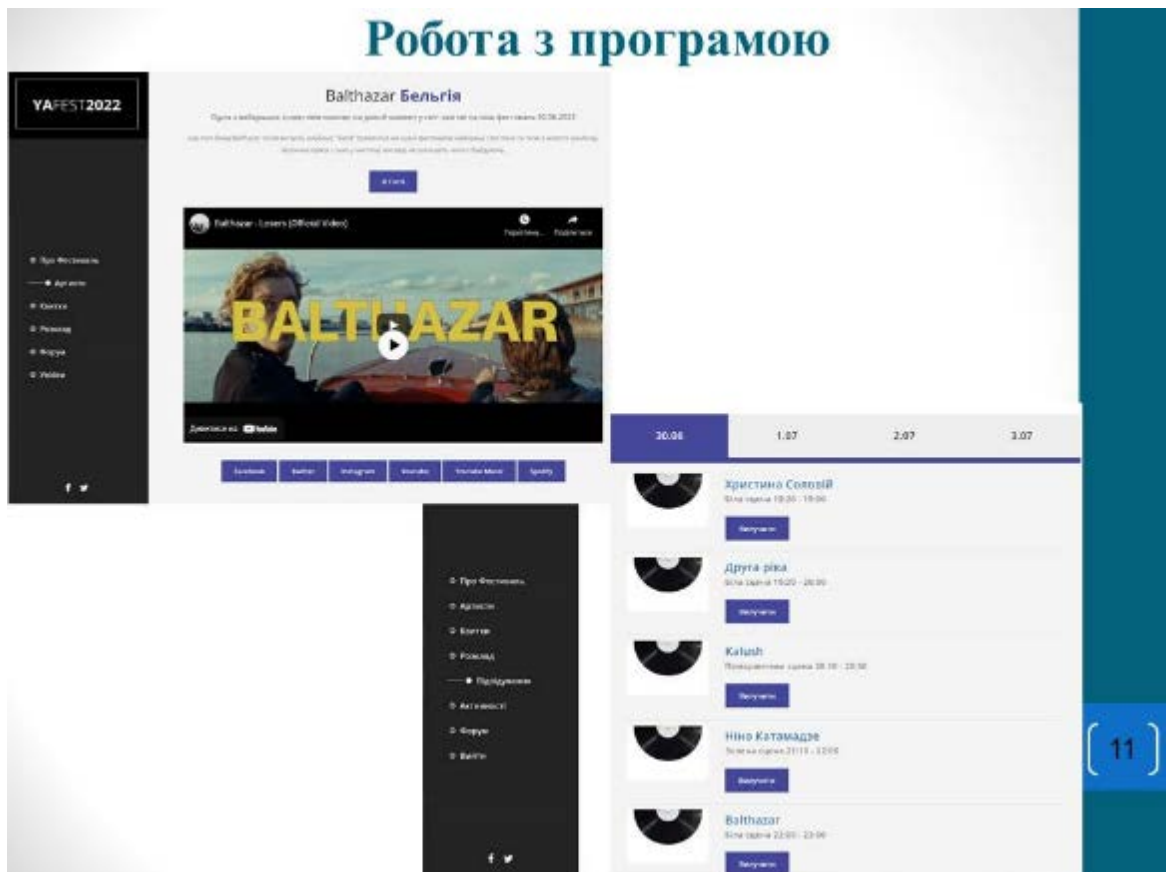


Рисунок В.11 – Слайд 11

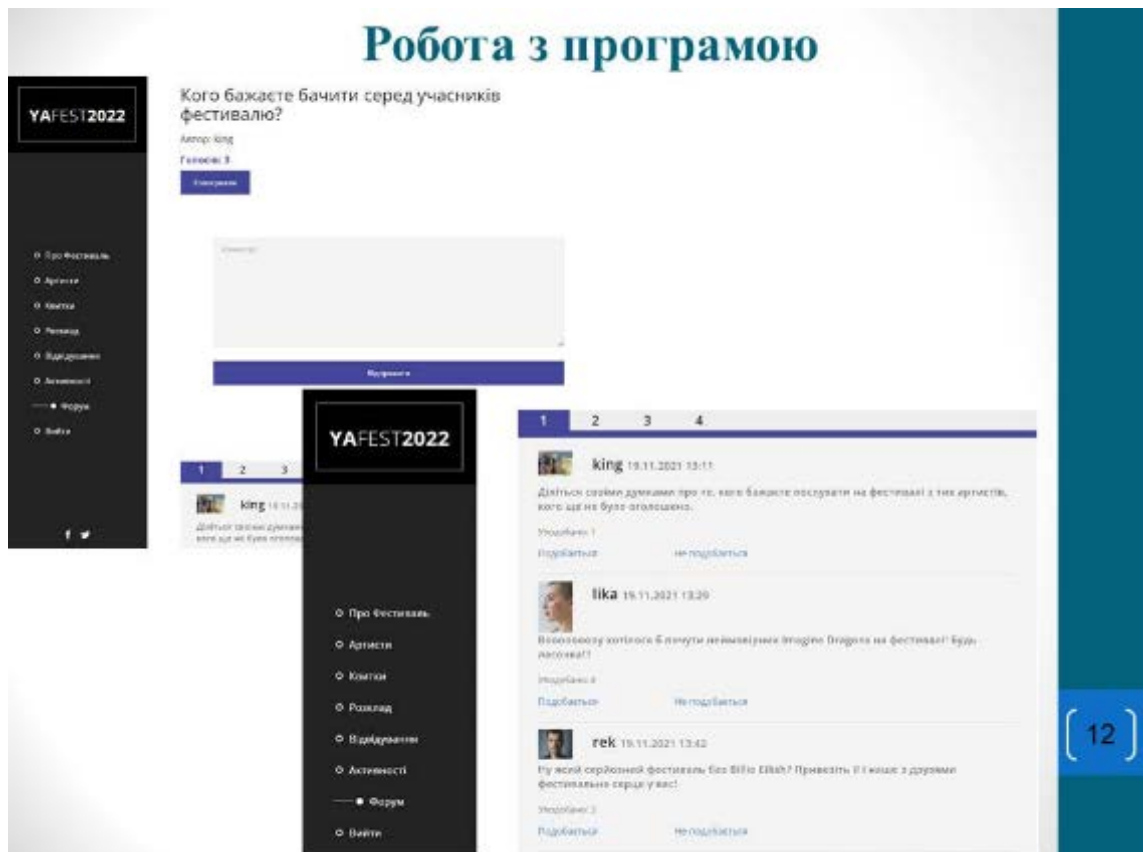


Рисунок В.12 – Слайд 12

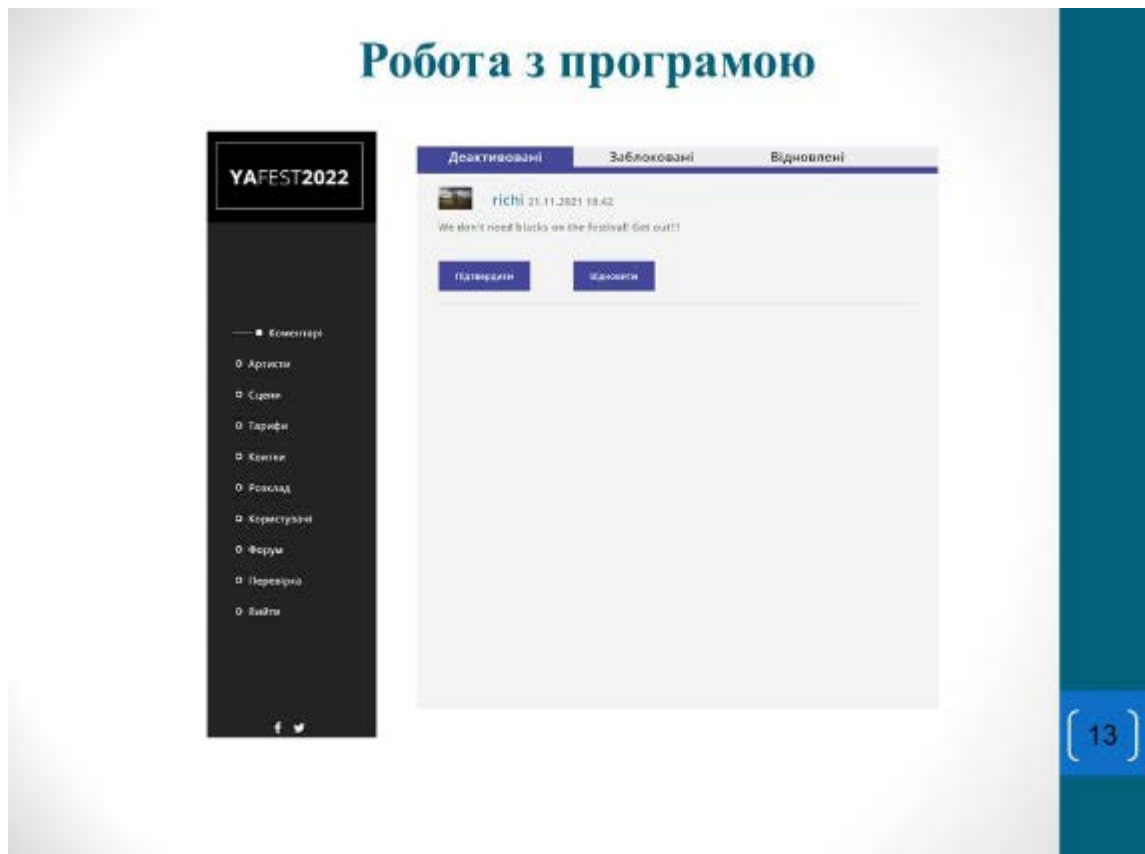


Рисунок В.13 – Слайд 13

Результати експериментального дослідження

Модель / метод	Точність, %
Наївний байєсів класифікатор	54,1
Запропонований метод керування коментарями відвідувачів музичного фестивалю	86,3

14

Рисунок В.14 – Слайд 14

Висновки

Наукова новизна роботи полягає в запропонованому методі керування коментарями відвідувачів музичного фестивалю, який ґрунтується на виконанні оброблення коментаря користувача моделлю GloVe з подальшою класифікацією методом Random Forest, що дозволяє визначати керуючі рішення з метою уникнення репутаційних втрат для фестивалю.

Визначено функціональні вимоги до програмного забезпечення, виконано проектування бази даних зі створенням її структури, визначено структуру програми. За результатами проектування виконано програмну реалізацію зі створенням відповідного набору класів.

Розроблене програмне забезпечення дозволяє переглядати інформацію про артистів, які беруть участь у фестивалі, розклад їх виступів, купувати квитки на фестиваль, створювати власний розклад відвідування фестивалю, переглядати і створювати теми у форумі, публікувати і переглядати коментарі в темах, оцінювати теми, уподобувати коментарі. Експериментальне дослідження підтвердило, що запропонований метод дозволяє отримати придатну для використання на практиці точність класифікації коментарів.

Усі поставлені завдання на роботу виконано.