Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Комп'ютерна система табелювання робочого часу на основі RFID-технології*

Виконав(ла): студент(ка) *IV* курсу, групи *CIс-41*
спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

	(підпис)	*Мельник Н. О.* (прізвище та ініціали)
Керівник	(підпис)	*Тиш Є. В.* (прізвище та ініціали)
Нормоконтроль	(підпис)	*Луцик Н. С.* (прізвище та ініціали)
Завідувач кафедри	(підпис)	*Осухівська Г. М.* (прізвище та ініціали)
Рецензент	(підпис)	*Муж В. В.* (прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Осухівська Г.М.
(підпис) (прізвище та ініціали)
« ____ » _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Мельнику Назарію Олександровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютерна система табелювання робочого часу на основі RFID-технології*

Керівник роботи *Тиш Євгенія Володимирівна, к.т.н., доцент каф. КС*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*28*» *02* *2023* року № *4/7-237*

2. Термін подання студентом завершеної роботи *15.06.2023 р.*

3. Вихідні дані до роботи *Технічне завдання*

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналіз технічного завдання
2. Проектна частина
3. Практична частина
4. Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Функціональна схема
2. Структурна схема
3. Блок-схема алгоритму програми
4. Електрична принципова схема

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,	Пилипець М. І., д. т. н.,		
основи охорони праці	професор кафедри МТ		

7. Дата видачі завдання *21.01.2023 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	*Розробка та затвердження технічного завдання*	*18.01-21.01*	*Виконано*
2	*Аналіз технічного завдання та обґрунтування можливих*	*22.01-20.02*	*Виконано*
	рішень		
3	*Розробка структурної та функціональної схеми*	*21.02-19.03*	*Виконано*
4	*Розробка електричної принципової схеми, вибір*	*20.03-20.04*	*Виконано*
	елементної бази		
5	*Розробка програмного забезпечення для проектованої*	*21.04-15.05*	*Виконано*
	системи		
6	*Опрацювання питань розділу «Безпека*	*16.05-28.05*	*Виконано*
	життєдіяльності, основи охорони праці»		
7	*Оформлення пояснювальної записки кваліфікаційної*	*29.05-05.06*	*Виконано*
	роботи бакалавра		
8	*Оформлення графічної частини*	*06.06-11.06*	*Виконано*
9	*Попередній захист кваліфікаційної роботи бакалавра*	*12.06-18.06*	*Виконано*
10	*Захист кваліфікаційної роботи бакалавра*	*19.06-23.06*	

Студент _____ *Мельник Н. О.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Тиш Є. В.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Комп’ютерна система табелювання робочого часу на основі RFID-технології // Кваліфікаційна робота бакалавра // Мельник Назарій Олександрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних систем та мереж, група СІс-41 // Тернопіль, 2023 // с. – 126, рис. – 38, табл. – 5, аркушів А1 – 4, додат. – 5, бібліогр. – 21.

Ключові слова: КОМП’ЮТЕРНА СИСТЕМА, ТАБЕЛЮВАННЯ РОБОЧОГО ЧАСУ, RFID, МІКРОКОНТРОЛЕР, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Кваліфікаційну роботу бакалавра присвячено розробці комп’ютерної системи для табелювання робочого часу працівників на підприємстві, яку реалізовано на основі безконтактної технології RFID.

На основі результатів огляду та аналізу сучасних апаратних і програмних систем табелювання робочого часу розроблено функіональну та структурну схему комп’ютерної системи табелювання робочого часу на основі RFID-технології.

Здійснено обґрунтування вибору елементної бази системи та описано процес розробки електричної принципової схеми апаратної частини на основі вибраної елементної бази.

Розроблено алгоритм роботи апаратної частини системи та здійснено опис програмних функцій мікроконтролера.

Описано розробку програмного забезпечення серверної частини, яке включає в собі бекенд та фронтенд частини, а також використання реляційної бази даних.

Розглянуто основні питання безпеки життєдіяльності та основ охорони праці, стосовно проєктованої системи та її використання.

ABSTRACT

Computer system of timesheet based on RFID technology // Qualification work for the bachelor`s degree // Melnyk Nazarii Oleksandrovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Computer Systems and Networks Department, Group CIs-41 // Ternopil, 2023 // p. – 126, fig. – 38, tab. – 5, sheets A1 – 4, additions. – 5, ref. – 21.

Keywords: COMPUTER SYSTEM, TIMESHEET, RFID, MICROCONTROLLER, SOFTWARE.

The bachelor's qualification work is devoted to the development of a computer system for timesheeting of employees at the enterprise, which is implemented on the basis of contactless RFID technology.

On the basis of the results of the review and analysis of modern hardware and software timesheeting systems, a functional and structural diagram of a computer-based timesheeting system based on RFID technology was developed.

The selection of the element base of the system is substantiated and the process of developing the electrical schematic diagram of the hardware part based on the selected element base is described.

An algorithm for the operation of the hardware part of the system was developed and a description of the software functions of the microcontroller was carried out.

The development of server-side software, which includes back-end and front-end parts, as well as the use of a relational database, is described.

The main issues of life safety and the basics of labor protection, in relation to the designed system and its use, are considered.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

API – Application Programming Interface;

DAO – Data Access Object;

EEPROM – Electrically Erasable Programmable Read-Only Memory;

GPIO – General-Purpose Input/Output;

HTTP – HyperText Transfer Protocol;

I2C – Inter-Integrated Circuit;

I2S – Integrated Inter-chip Sound;

IDE – Integrated Development Environment;

JDK – Java Development Kit;

JSP – Java Server Page;

LCD – Liquid Crystal Display;

MVC – Model-View-Controller;

NTP – Network Time Protocol;

ORM – Object-Relational Mapping;

RFID – Radio Frequency IDentification;

RGB – Red, Green, Blue;

ROM – Read-Only Memory;

SPI – Serial Peripheral Interface;

SRAM – Static Random Access Memory;

UART – Universal Asynchronous Receiver/Transmitter;

UID – User IDentifier;

USB – Universal Serial Bus;

Wi-Fi – Wireless Fidelity;

АЦП – Аналого-Цифровий Перетворювач;

СКБД – Система Керування Базами Даних;

ЦАП – Цифро-Аналоговий Перетворювач;

ШІМ – Широтно-Імпульсна Модуляція.

ВСТУП

Один з важливих аспектів будь-якого підприємства у їх діяльності – це облік робочого часу. Лише з допомогою обліку робочого часу є можливість дізнатись, коли і скільки часу робітник знаходився на роботі. На сьогоднішній день існує дуже багато засобів обліку робочого часу. Найбільш поширений зі всіх – табелювання обліку робочого часу.

Робочий час – це час, протягом якого працівники мають перебувати на власному робочому місці та виконувати свої трудові обов'язки. Також окрім цього є такий важливий аспект, як режим робочого часу. Він визначає точний час початку, завершення робочої зміни, різного роду перерв та неробочих днів.

Законодавством на юридичну особу, яка наймає працівників, покладено обов'язок організації обліку робочого часу. Саме тому зазвичай на підприємстві дана юридична особа, або ж відповідальна за облік особа, веде табель обліку робочого часу.

Табель обліку робочого часу – це документ, який вміщує в собі використання робочого часу кожного з працівників, яких було найнято на підприємство, незалежно від того чи на постійній основі, чи на тимчасовій. Дані про фактично відпрацьований, а також невідпрацьований час містяться саме у табелі обліку робочого часу. Дані табеля можна застосувати для підтвердження стажу працівника, аналізу ефективності працівника та структури підприємства в загальному [2, 3].

Таким чином, табель обліку робочого часу – це такий документ, у якому враховано фактично відпрацьований працівником час. Окрім цього даний документ часто виступає первинним при нарахуванні заробітної плати працівників. Табель складається кожного місяця та передається бухгалтерії, де, відповідно, проводяться нарахування працівникам їхньої заробітньої плати, яка залежить від кількості фактично відпрацьованих ними робочих годин.

На сьогоднішній день для більш точного табелювання обліку робочого часу працівників все частіше використовуються комп’ютеризовані системи табелювання [4, 5]. Такі системи у значній мірі сприяють більш точному обліку, особливо коли юридична особа (або відповідальна за облік особа) практично немає можливості табелювати працівників. При цьому правове положення працівника не погіршується. Електронна система тільки сприяє належному виконанню обов’язків, а також дотриманню дисципліни праці на підприємстві.

На ринку представлено немало виробників систем табелювання обліку робочого часу, які пропонують власні вироби з великими функціональними можливостями. Проте, вартість сучасної техніки є досить високою, тому розробка подібної системи з доступною ціною є актуальною задачею.

Метою даної кваліфікаційної роботи є розробка комп’ютерної системи для табелювання робочого часу на малих та середніх підприємствах на основі технології RFID.

Для цього необхідно вирішити низку задач:

– здійснити огляд та критичний аналіз існуючих аналогів;

– створити функціональну та структурну схеми системи для табелювання робочого часу на підприємстві;

– здійснити вибір необхідних елементів для реалізації системи;

– розробити сайт, на якому будуть відображатись табель обліку робочого часу та функції керування системою;

– написати відповідне програмне забезпечення для реалізації усіх необхідних функціональних можливостей проєктованої системи.

РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

В даному розділі кваліфікаційної роботи виконано аналіз вимог до комп'ютерної системи табелювання робочого часу на основі технології RFID. Проведено критичний огляд та аналіз існуючих систем схожого типу. Виявлено їх позитивні та негативні сторони.

1.1 Аналіз вимог до комп'ютерної системи

Комп'ютерна система табелювання робочого часу на основі RFID з технічної точки зору – це сукупність засобів, призначених для ідентифікації працівників та ведення обліку робочого часу даних працівників. Область застосування – для використання на підприємстві. Тип системи – автоматизована система із можливістю ручного керування.

Під терміном «табелювання робочого часу» (далі табелювання) називається запис робочих годин, які працівник відпрацював за будь-який конкретний обліковий період. Табелювання фіксує «кредитні» та «дебетові» години. Перші нараховуються коли працівник працює більше тих годин, які було встановлено його трудовим чи колективним договорами. Другі ж нараховуються тоді, коли працівник працює менше. Ведення табелювання покликане забезпечити гнучкість в управлінні робочим часом, що приносить користь як роботодавцям, так і працівникам. Роботодавці можуть отримати вигоду, маючи можливість закликати працівників працювати більше, ніж їхні договірні години, в той час як працівники можуть накопичувати відпрацьовані години, щоб в подальшому забезпечити собі більш тривалі відпустки [2].

Таким чином виникає необхідність до підвищення функціональних можливостей систем табелювання, таких як надійність та точність отриманих і оброблених даних, придатність до ремонту, швидкість реакції системи на зміни, зручність обслуговування як апаратної частини, так і програмної.

Відповідно до затвердженого технічного завдання, необхідно розробити систему, яка відповідатиме сучасним досягненням інформаційних технологій мікроелектроніки та програмного забезпечення і виконуватиме наступні функції:

– ідентифікацію працівника з допомогою технології RFID;

– фіксація точного часу доби в моменти ідентифікації працівників;

– передача оброблених даних на сервер;

– табелювання сервером отриманих даних;

– відображення табелів кожного з працівників на сайті у вигляді таблиць.

Табелювання повинне вестись на всіх працівників, у тому числі тих, які прийняті тимчасово.

1.1.1 Аналіз вимог до компонентів апаратного забезпечення

Однією з головних вимог технічного завдання є забезпечення автоматизованої роботи комп’ютерної системи на основі технології RFID. Для цього в проєктованій системі необхідно передбачити модуль RFID, який зчитуватиме інформацію про працівників завдяки їх персональним RFID-міткам та передаватиме мікроконтролеру. У свою чергу мікроконтролер повинен обробити ці дані та надіслати їх на сервер і для цього на його борті необхідний модуль Wi-Fi. Також комп’ютерна система має мати світлодіодну індикацію для індикації її роботи та стану. Доступ модуля Wi-Fi до сервера здійснюється через маршрутизатор.

Відповідно до вимог, що регламентовані відповідними стандартами, визначаються характеристики сервера та ширина каналу лінії доступу до мережі Інтернет.

1.1.2 Аналіз вимог до компонентів програмного забезпечення

До програмного забезпечення входять:

– програма мікроконтролера;

– сервер;

– мови програмування, які інтерпретуються браузером та сервером.

На основі мов програмування розробляється вебсайт, який складається з «передньої частини» («Front End») та «задньої частини» («Back End»).

Атрибутами програмного забезпечення повинні бути:

– ясність;

– зручність внесення змін;

– практичність;

– адекватність;

– функціональність.

Зовнішні інтерфейсні вимоги вебсайту:

– зручність у використанні;

– зрозумілий та простий у використанні інтерфейс;

– коректність роботи.

Програмне забезпечення мікроконтролера повинне давати не більше 2 помилок за добу. Сайт повинен давати не більше 2 помилок в місяць.

Серед логічних вимог до програмного забезпечення комп’ютерної системи є:

– робота бази даних та керування нею повинні бути коректними;

– дані, надіслані апаратними компонентами системи не повинно бути спотворено при збереженні їх у базу даних чи відображенні на сайті.

Функції керування системою повинні бути доступними за логіном та паролем конкретного зареєстрованого користувача в будь-який 30-денний період.

1.2 Аналіз можливих рішень поставленого завдання

Системи табелювання робочого часу на підприємстві поділяються на програмні та апаратно-програмні.

Серед програмних систем особливо популярними є системи автоматичного стеження. Такі системи проводять відстеження запусків кожної програми на комп'ютері, перехоплюють повідомлення у месенджерах або ж виконується моніторинг натискання клавіш на клавіатурі. Окрім цього існують системи, які роблять скріншоти чи відеозапис робочого столу на комп'ютерах працівників. Проте такі системи, які виконують контроль робочого часу працівника порушують статті 182 та 163 Кримінального кодексу України і роботодавець спочатку зобов'язаний отримати письмову згоду працівника, щоб мати можливість використовувати ці системи.

Натомість є можливість використовувати особисті та колективні системи добровільного обліку витрат часу, які називаються тайм-трекерами [5]. Дані програмні засоби для ведення обліку робочого часу цілком законні.

Основною проблемою програмних систем табелювання робочого часу є те, що ці системи можуть використовуватись тільки на офісних підприємствах, в яких у кожного з працівників наявне власне робоче місце з персональним комп'ютером.

Основна перевага апаратно-програмних систем обліку робочого часу проти програмних в тому, що перші не мають проблеми розташування других і можуть бути розміщені у точці реєстрації співробітників.

Апаратно-програмні системи забезпечують високу автоматизацію, адже дозволяють легко табелювати робочий час, для цього працівнику необхідно лиш прикласти свій відбиток пальця до біометричного сканера або

ідентифікаційну карту до відповідного модуля зчитування. Також подібні системи у своєму комплекті поставки мають також спеціалізовані програми, в яких і ведеться табелювання робочого часу усіх працівників.

Системи табелювання робочого часу апаратно-програмного типу [4] зазвичай працюють наступним чином: персонал, прийшовши на роботу, ідентифікує себе одним з методів, закладених у функціоналі системи (наприклад зчитування відбитку пальця, контурів лиця, ідентифікаційної карти тощо), після чого система фіксує час приходу або виходу з роботи. Ці події фіксуються в автономній пам'яті системи, після чого ці дані передаються на комп'ютер/сервер через Ethernet, Wi-Fi, USB тощо. Після передачі дані записуються і формують табелі робочого часу працівника. Особливо ефективною є передача даних про події через мережеві технології, що дозволяє виконувати контроль роботи працівників навіть на віддалених філіях підприємства.

Апаратно-програмні системи табелювання також зазвичай мають функцію контролю доступу, наприклад, можливість керування електронними замками (такими як електрозащіпки, електромагнітні замки та турнікети). У цих випадках система при реєстрації події одночасно подає відповідний цифровий сигнал. Головним недоліком апаратно-програмних систем є те, що на ринку ці товари є доволі дороговартісними.

Для сортування та зручного керування даними про облік робочого часу системами табелювання використовуються системи керування базами даних (СКБД). СКБД – це набір взаємопов'язаних даних (баз даних), а також програм для керування доступом до цих них.

1.3 Огляд існуючих систем табелювання робочого часу на підприємстві

На даний момент на ринку представлено багато варіантів систем табелювання робочого часу на підприємстві. Розглянуто та проаналізовано найпоширеніші з цих рішень.

1.3.1 Термінал контролю доступу та обліку робочого часу Dahua DHI-ASA1222G

DHI-ASA1222G – це апаратно-програмна система, що призначена для контролю доступу. Додатково має функції для обліку робочого часу. Розроблена всесвітнім виробником систем охорони Dahua. Доступ, в залежності від встановленого налаштування, надається за скануванням відбитка пальця, картою доступу або введенню попередньо встановленого пароля [4]. Зовнішній вигляд системи зображено на рис. 1.1.

Рисунок 1.1 – Зовнішній вигляд системи DHI-ASA1222G

Термінал має на своєму борті 32-бітний процесор та вбудований рідкокристалічний дисплей. Довжина дисплею – 7 дюймів. Окрім цього термінал доповнено пам’яттю ROM – 16 MB, SRAM – 8 MB. Дозволяє зберігати у своїй пам’яті не більше 1000 записів.

Головним призначенням системи Dahua DHI-ASA1222G є відкриття доступу до приміщень на підприємствах та офісних будівлях. Окрім цього, вона також дозволяє виконувати облік робочого часу працівників, тобто фіксує час, який людина проводить на роботі та на обіді, а також коли вона приходить і залишає власне робоче місце. Завдяки такому функціоналу система дозволяє підвищити ефективність праці на підприємстві.

Dahua DHI-ASA1222G підтримує фіксований, ручний/автоматичний та примусовий режими відвідуваності.

Основні переваги даної системи:

– можливість імпорту/експорту записаних даних на комп'ютер/сервер через інтерфейс USB 2.0 або з допомогою технології «Ethernet»;

– низьке споживання енергії;

– отримання доступу та табелювання робочого часу трьома різними методами – за відпечатком пальця, за картою доступу або за паролем. Наявна можливість комбінувати ці методи;

– малий час відгуку системи на ідентифікацію користувача за картою доступу або відбитком пальця (0.1 с. та 1.5 с. відповідно).

– підтримка різних режимів роботи;

Незважаючи на переваги даної системи, вона має три основні недоліки, а саме:

– неможливість керувати системою віддалено;

– регулярне ручне вивантаження даних про робочий час працівників на окремий комп'ютер/сервер;

– висока вартість системи, що є найбільш вагомим недоліком для використання на малих підприємствах.

1.3.2 Програмний продукт трекінгу робочого часу Time Analytics

Серед програмних систем одним з найкращих є Time Analytics. Дана система найкраще підходить для відстеження часу та ведення табелів бухгалтерського обліку робочого часу працівників, оподаткування, аудиту та консультування малого і середнього бізнесу [5]. Зовнішній вигляд початкової сторінки Time Analytics зображено на рис. 1.2.

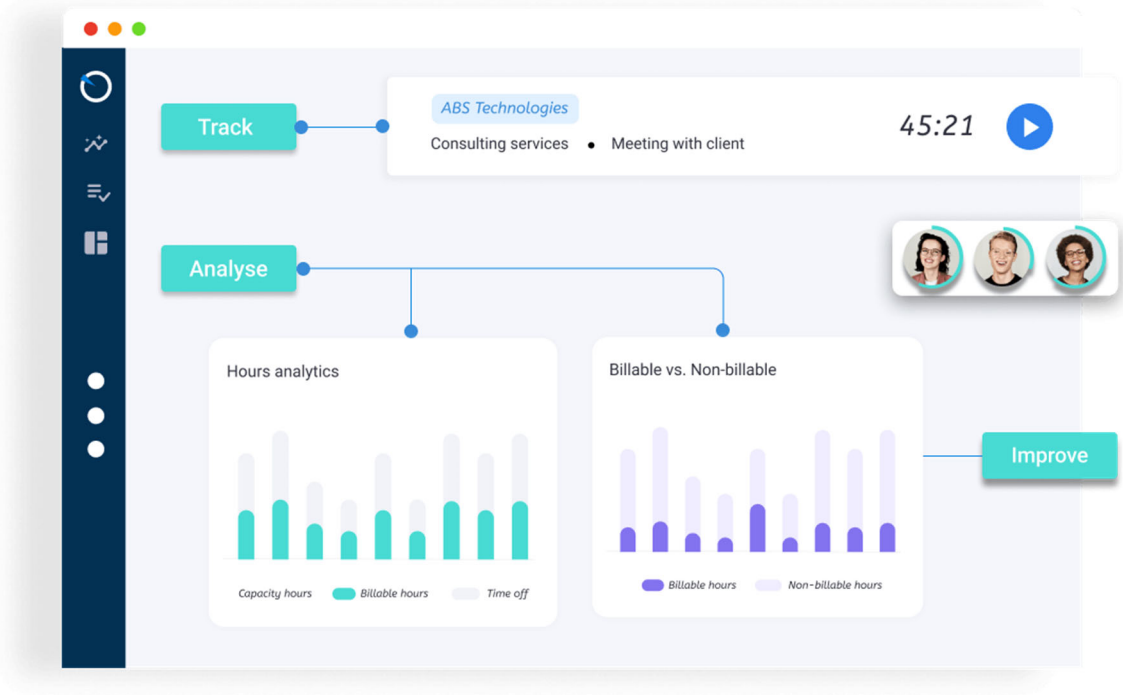

Рисунок 1.2 – Початкова сторінка Time Analytics

Основні функціональні можливості даної програмної системи:

– Відстеження часу. Дана функція допомагає працівникам відстежувати свій час і підвищувати продуктивність.

– Табелювання робочого часу. Система складає докладні щоденні табелі працівників, у яких відображається кожен запис часу для проєкту, клієнта та завдання. Є можливість перегляду табелів та їх редагування менеджером, а також функції фільтрування та експорту табеля в програму Excel.

– Звітність про час. Система надає працівникам / менеджеру інформацію про погодинну ставку доходу на одного клієнта, дозволяє переглядати за тим, яка лінія послуг є найбільш вигідною для компанії.

– Звітування про продуктивність та ефективність команди. Time Analytics надає звітність про кредитні та дебетові години для кожного працівника в команді, звітність про використання часу та статистику про понаднормові години [5].

Основними недоліками даної програмної системи є:

– щомісячна оплата наданих виробником послуг;

– можливість використання тільки на підприємствах, у яких кожен з працівників має власне робоче місце з персональним комп’ютером, що в загальному сильно обмежує її використання.

В результаті аналізу літературних джерел встановлено, що для системи табелювання робочого часу важливі такі аспекти, як швидкість роботи, точність, функціональність, використання сучасних технологій та інтерфейсів, таких як Ethernet, USB, а також можливість використання системи на різних видах підприємств. Проаналізовано сучасні системи табелювання робочого часу та встановлено недоліки, які обмежують їх використання.

РОЗДІЛ 2 ПРОЄКТНА ЧАСТИНА

2.1 Розробка узагальненої структури комп'ютерної системи табелювання робочого часу на основі технології RFID

На початковому етапі проєктування комп'ютерної системи табелювання робочого часу на основі технології RFID розроблено узагальнену структурну схему системи. Структурна схема призначена для відображення головних функціональних компонентів та зв'язків між ними. Окрім цього структурна схема в загальному відображає призначення системи, її елементів та їхні з'єднання. Використовується для загального ознайомлення із комп'ютерною системою.

Структурну схему комп'ютерної системи табелювання робочого часу зображено на рис. 2.1.

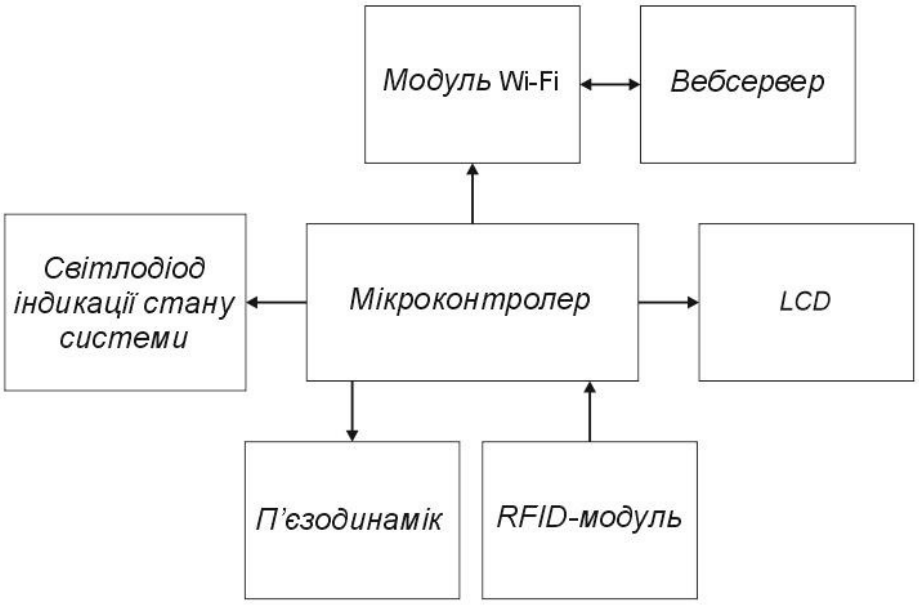

Рисунок 2.1 – Структурна схема комп'ютерної системи табелювання робочого часу на основі технології RFID

Позначення функціональних елементів системи на структурній схемі:

– мікроконтролер – призначений для прийому, обробки та надсилання даних. Виконує керування апаратною частиною системи;

– LCD – рідкокристалічний дисплей, призначений для відображення системної інформації;

– RFID-модуль – модуль, призначений для зчитування RFID-міток та надсилання зчитаних даних мікроконтролеру;

– п’єзодинамік – призначений для звукового сповіщення про успішне зчитання мітки;

– світлодіод індикації стану системи;

– модуль Wi-Fi – призначений для передачі оброблених мікроконтролером даних на сервер;

– сервер – вузол, на який надсилаються дані табелювання робочого часу та на якому розташовано вебсайт, з допомогою якого можна переглянути ці дані. Доступ до вебсайту надається через локальну мережу.

2.2 Обґрунтування вибору апаратного забезпечення комп’ютерної системи табелювання робочого часу на основі технології RFID

Вибір елементної бази апаратного забезпечення проєктованої системи проведено відповідно до структурної схеми, зображеної на рис. 2.1.

2.2.1 Обґрунтування вибору мікроконтролера

Систему створено на основі мікроконтролера ESP32 DevKit V1 [6, 7]. Даний мікроконтролер обрано через те, що він має достаню потужність, кількість пінів вводу-виводу, усі необхідні апаратні інтерфейси та технології, включаючи модулі Bluetooth і Wi-Fi, а також дає можливість вдосконалювати проєктовану систему у майбутньому шляхом підключення додаткових модулів (наприклад, модуля біометричного сканера відбитків пальців).

Апаратну частину IoT-платформи виконано на модулі ESP-WROOM-32 з однокристальним чіпом ESP32-D0WDQ6 виробництва компанії Espressif [8]. Дана серія мікропроцесорів ESP32 містить на своєму борті інтегровані контролери Wi-Fi та Bluetooth. Мікроконтролери ESP32 використовують мікропроцесор Tensilica Xtensa LX6 в двоядерних та одноядерних варіаціях. Вони також включають в себе модулі керування живленням, антенні перемикачі, приймач з низьким рівнем шумів, підсилювач потужності та фільтри.

Зовнішній вигляд плати ESP32 зображено на рис. 2.2.

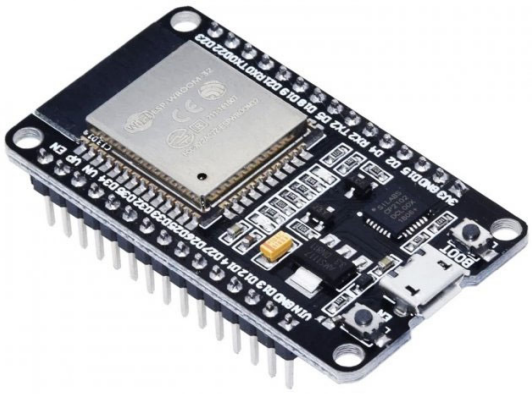

Рисунок 2.2 – Зовнішній вигляд плати ESP32 DevKit V1

Технічні характеристики чіпа ESP32-D0WDQ6 наведено у таблиці 2.1.

Таблиця 2.1 – Технічні характеристики чіпа ESP32-D0WDQ6

Параметр	Значення
Тип монтажу	Поверхневий (технологія SMT)
Ядро	Tensilica Xtensa LX6
Вихідна потужність	20.5 дБМ
Кількість виводів вводу-виводу	34 I/O
Кількість каналів АЦП	18 каналів
Кількість таймерів	2 таймери
Робоча температура	Від -40 °C до +125 °C
Максимальна швидкість передачі	150 Мб / с

Продовження таблиці 2.1

Максимальна тактова частота	240 МГц
Робоча напруга живлення	від 2.3 В до 3.6 В
Розмір програмованої пам’яті (EEPROM)	448 кБ
Розширення АЦП	12
Тип ОЗП	SRAM
Розмір даних ОЗП	16 кБ, 520 кБ
Розрядність шини даних	32 біти

Зовнішній вигляд чіпа ESP32-D0WDQ6 зображено на рис. 2.3.

Рисунок 2.3 – Зовнішній вигляд чіпа ESP32-D0WDQ6

Технічні характеристики мікроконтролера ESP32 наведено у таблиці 2.2.

Таблиця 2.2 – Технічні характеристики мікроконтролера ESP32

Параметр	Значення
Процесор	Tensilica Xtensa LX6
Робоча напруга	3.3 В
Тактова частота	240 мГц
Робочий струм	80 мА в середньому
Діапазон частот	2.4-2.5 ГГц
Статична оперативна пам’ять (SRAM)	520 КБ
Флеш-пам’ять	4 МБ
Цифрові виходи введення / виведення (GPIO)	21 (16 з яких можуть працювати в режимі ШІМ)
Аналогові входи АЦП (12 біт)	15
Аналогові виходи ЦАП (8 біт)	2
Тактова частота	16 МГц

У ESP32 кристал включає 2-ядерний 32-розрядний процесор Tensilica Xtensa LX6, а також 520 Кб пам'яті SRAM і 448 Кб флеш-пам'яті, 4 Мб зовнішньої флеш-пам'яті. Залежно від режиму споживання енергії тактову частоту можна встановити до 240 МГц.

Наявні вбудовані температурний датчик, датчик Холла, інфрачервоний контролер на приймання та передачу, контролер сенсорних кнопок, Bluetooth, Wi-Fi (відповідає стандартам 802.11b, 802.11g та 802.11n).

Мікросхема CP2102 містить перетворювач UART-USB. Даний модуль необхідний для забезпечення зв'язку модуля мікроконтролера із USB-портом комп'ютера.

Для завантаження скетчу та живлення модуля ESP32 DevKit V1 через персональний комп’ютер призначений роз’єм micro-USB. Однією з переваг модуля є низьке споживання електроенергії і гнучкий вибір різних режимів роботи. Режим deep sleep mode дозволяє зменшити споживання сили струму до 20 мкА [7].

На пінах вводу-виводу можна сконфігурувати апаратні інтерфейси:

– 3xUART;

– 3xSPI;

– 2xI2C;

– 3xI2S.

Виводи живлення:

– VIN – вивід, до якого можна підключити зовнішнє джерело живлення. Діапазон напруг для підключення до даного виводу від 5 до 14 вольт;

– 3V3 – вивід від стабілізатора напруги з виходом 3,3 вольти. Максимальна сила струму – 1 ампер;

– GND – виводи «земля».

Універсальний асинхронний приймач-передавач (UART) — це апаратний інтерфейс, який дозволяє приймати та передавати дані між компонентами комп'ютерів / контролерів і периферійними пристроями.

Мікросхема ESP32 має три контролери UART. Кожен з них має ідентичний набір регістрів для спрощення програмування, а також забезпечення більшої гнучкості.

I2C — це послідовний, синхронний, напівдуплексний протокол зв'язку, який дозволяє співіснувати та комунікувати кільком головним і підлеглим пристроям на одній шині. Шина I2C складається з двох ліній: послідовної лінії даних (SDA) і послідовної синхронізації (SCL). Обидві лінії потребують підтягувальних резисторів.

Драйвер SPI Master — це програма, яка керує периферійними пристроями SPI ESP32, коли вони функціонують як головні. ESP32 дозволяє інтегрувати 4 периферійні пристрої, які працюють та комунікують через інтерфейс SPI.

Виводи плати ESP32 та їх позначення зображено на рис. 2.4.

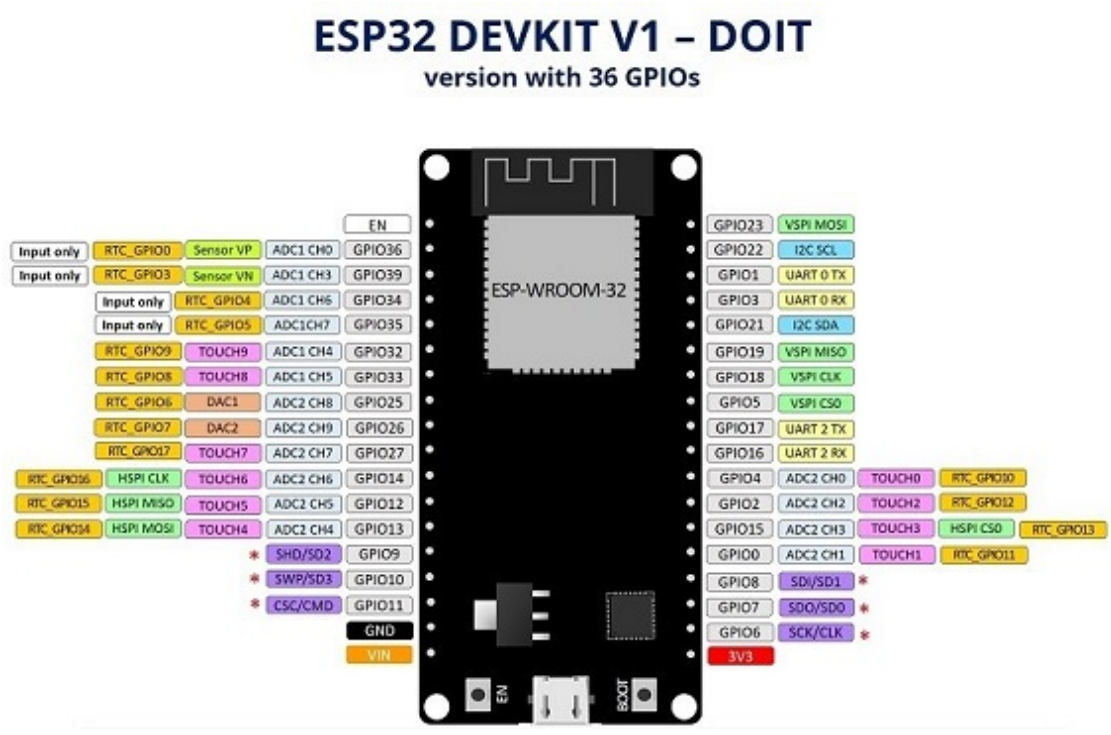

Рисунок 2.4 – Позначення виводів плати ESP32

Живлення на плату подається через роз’єм micro-USB або через вивід Vin. Напруга логічних рівнів на виводах плати - 3,3 В. Підключення пристроїв, які працюють від напруги живлення 5 В недопустиме, тому що це може призвести до пошкодження виводів або плати в цілому.

Умовне графічне позначення плати ESP32 на електричній принциповій схемі зображено на рис. 2.5.

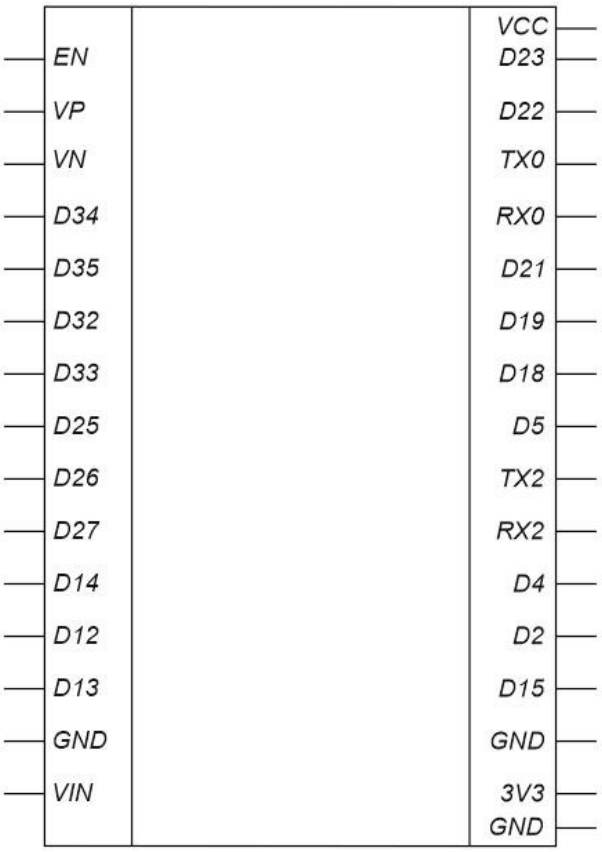

Рисунок 2.5 – Умовне графічне позначення ESP32 на електричній принциповій схемі

2.2.2 RFID-модуль RC522

Для ідентифікації працівників обрано технологію RFID та модуль RC522 [9], який працює на базі даної технології.

RFID (радіочастотна ідентифікація, англ. – Radio Frequency IDentification) – це технологія, що використовується для безконтактної ідентифікації об’єктів, використовуючи радіочастотні канали зв’язку.

Ідентифікація об’єктів проводиться за унікальним ідентифікатором, котрий має кожна електронна мітка.

Технологія RFID має наступні переваги:

– безконтактна – достатньо лише прикласти власну карту доступу;

– можливість прихованого встановлення міток;

– висока швидкість зчитування даних з міток та запису на них;

– функціонування у шкідливих середовищах;

– неможливість підробки міток, тому що вони мають унікальний ID.

Зовнішній вигляд модуля RC522 зображено на рис. 2.6.

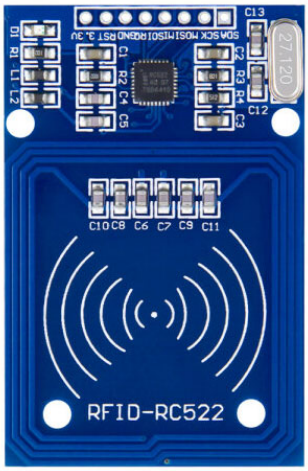

Рисунок 2.6 – Зовнішній вигляд модуля RC522

Модуль виконано на основі мікросхеми MFRC522. Дана мікросхема забезпечує роботу з мітками HF, робоча частота яких – 13,56 МГц [9].

Технічні характеристики RFID-модуля RC522 наведено у таблиці 2.3.

Таблиця 2.3 – Технічні характеристики RFID-модуля RC522

Параметр	Значення
Напруга живлення	3.3 В
Струм споживання	13-26 мА
Робоча частота	13.56 МГц
Дальність зчитування	0-60 мм
Інтерфейси	SPI, I2C, UART
Швидкість передачі даних	Максимальна 10 Мб / с

Позначення виводів модуля RC522 зображено на рис. 2.7.

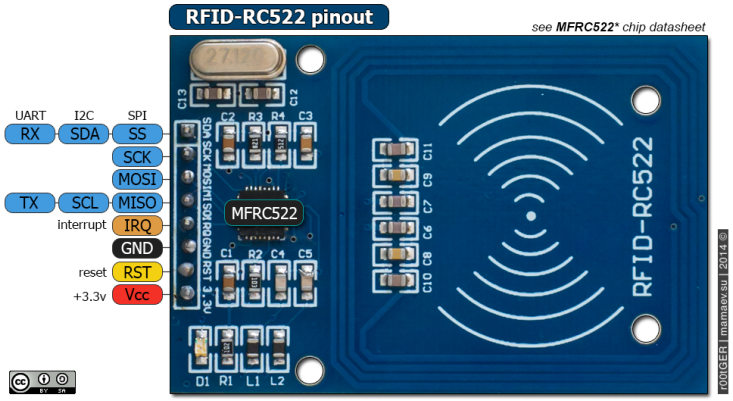

Рисунок 2.7 – Позначення виводів модуля RC522

Значення виводів модуля RC522 та їх з’єднання до ESP32:

– SS – сигнал ведучого – підключається до виводу D5;

– SCK – синхронізуючий сигнал – підключається до виводу D18;

– MOSI – передача від master до slave – підключається до виводу D23;

– MISO – передача від slave до master – підключається до виводу D19;

– RST – вивід скидання – підключається до виводу 3V3;

– IRQ – вивід переривання – не використовується;

– GND – земля – підключається до будь-якого виводу GND;

– Vcc – живлення 3.3 В – підключається до виводу 3V3.

Сигнал скидання RST – це сигнал, який необхідний для перезагрузки модуля. При отриманні логічного «0» на даний вивід, виконується перезагрузка модуля. Якщо ж на виводі RST встановлено логічну «1», то він переходить в робочий режим.

Умовне графічне позначення модуля RC522 на електричній принциповій схемі зображено на рис. 2.8.

Рисунок 2.8 – Умовне графічне позначення модуля RC522 на електричній принциповій схемі

2.2.3 LCD 1602

LCD 1602 (див. рис. 2.9) – це рідкокристалічний дисплей, який використовується у системі для відображення системного часу та стану системи. Приблизні розміри даного – 8x3 см. Закріплений на невеликій платі з контактами [10].

Основними перевагами даного дисплею є:

– низька вартість;

– чітке відображення символів;

– витримка перепадів температур;

– невеликі розміри.

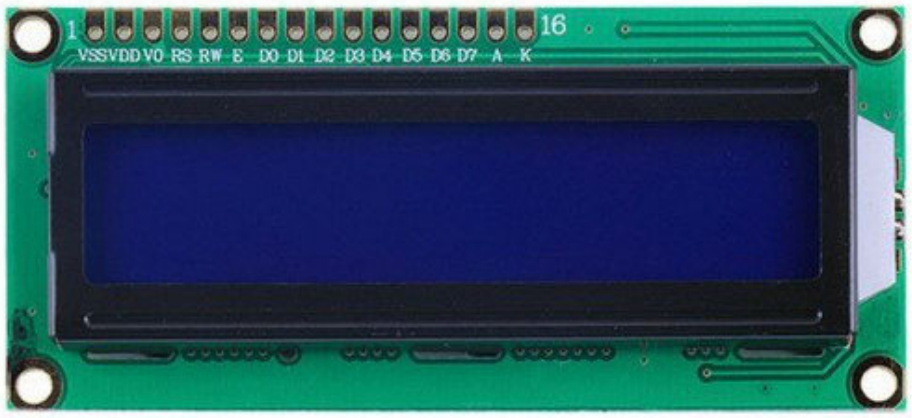

Рисунок 2.9 – Зовнішній вигляд LCD 1602

Значення виводів LCD 1602:

– GND – вивід «земля»;

– VDD – вивід живлення +5 В;

– VO – вивід для встановлення контрасту;

– RS – вивід «команди та дані»;

– RW – вивід «читання та запис»;

– E – вивід Enable;

– D0…D7 – лінії даних;

– A – анод підсвітки;

– K – катод підсвітки.

Технічні характеристики LCD 1602 наведено у таблиці 2.4.

Таблиця 2.4 – Технічні характеристики LCD 1602

Параметр	Значення
Тип відображення	Символьний
Тип підсвітки	Світлодіодна
Контролер	HD44780
Напруга живлення	+5 В
Формат символів	16x2
Діапазон робочої температури	Від -20 до +70 °C
Діапазон температури зберігання	Від -30 до +80 °C
Кут огляду	180°

Дані на дисплеї відображаються лише у форматі символів 16x2. Також на дисплеї наявна світлодіодна підсвітка, завдяки якій наявна можливість використовувати його у нічний час доби [10].

Умовне графічне позначення LCD 1602 на електричній принциповій схемі зображено на рис. 2.10.

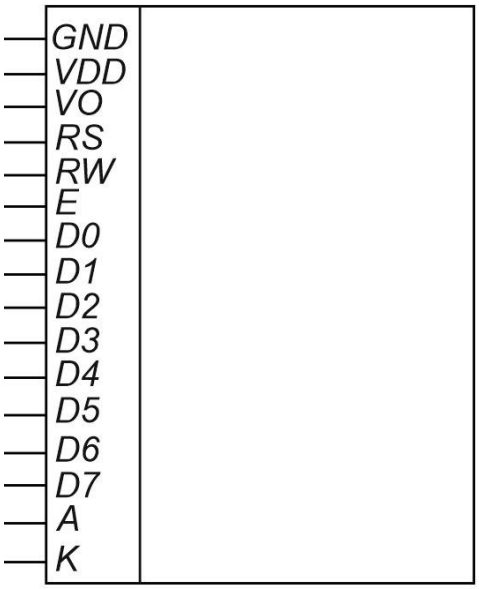

Рисунок 2.10 – Умовне графічне позначення LCD 1602 на електричній принциповій схемі

Так як у даного рідкокристалічного дисплея наявно 16 виводів, 6 із яких являються обов’язковими до підключення, проблема великої кількості контактів для підключення є найголовнішим недоліком. Для вирішення цієї проблеми використовується конвертер в I2C, зовнішній вигляд якого зображено на рис. 2.11.

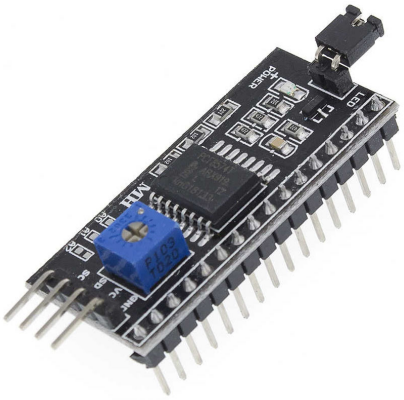

Рисунок 2.11 – Зовнішній вигляд конвертера

Умовне графічне позначення I2C-конвертера на електричній принциповій схемі зображено на рис. 2.12.

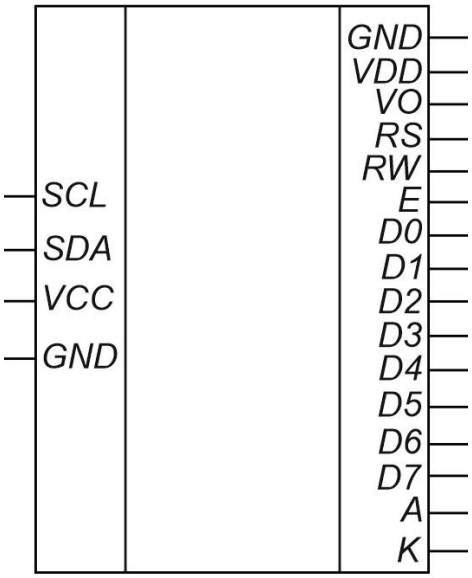

Рисунок 2.12 – Умовне графічне позначення I2C-конвертера на електричній принциповій схемі

З однієї сторони на платі конвертера наявна перша група виводів (16 ніжок), які відповідають стандартним виводам LCD 1602 та припаюються до них. З іншої сторони розміщено друга група виводів (4 ніжки), які з'єднуються безпосередньо до плати мікроконтролера ESP32:

– Вивід GND під'єднується до будь-якого виводу GND;
– Вивід VCC під'єднується до виводу Vin;
– Вивід SCL під'єднується до виводу D22;
– Вивід SDA під'єднується до виводу D21.

2.2.4 Активний п'єзодинамік KY-012

Для реалізації звукового сповіщення зчитування даних з RFID-мітки для комп'ютерної системи обрано активний п'єзодинамік KY-012 [11]. Принцип його роботи полягає у деформації шару п'єзоелектрика, зменшуючи та збільшуючи таким чином відстань до мембрани. Напруга живлення даного п'єзодинаміка 3,3 В, а частота звуку, яку він видає – 2 кГц.

Зовнішній вигляд п'єзодинаміка зображено на рис. 2.13.

Рисунок 2.13 – Зовнішній вигляд п’єзодинаміка

Умовне графічне позначення п’єзодинаміка на електричній принциповій схемі зображено на рис. 2.14.

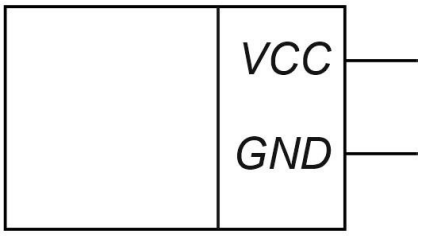

Рисунок 2.14 – Умовне графічне позначення п’єзодинаміка на електричній принциповій схемі

2.2.5 Світлодіод індикації SKV149

Для індикації поточного стану системи використовується RGB-світлодіод (спільний катод) довжиною 5мм марки SKV149 [12]. Зовнішній вигляд RGB-світлодіода зображено на рис. 2.15.

Рисунок 2.15 – Зовнішній вигляд RGB-світлодіода SKV149

Позначення виводів світлодіода та його розміри у міліметрах зображено на рис. 2.16.

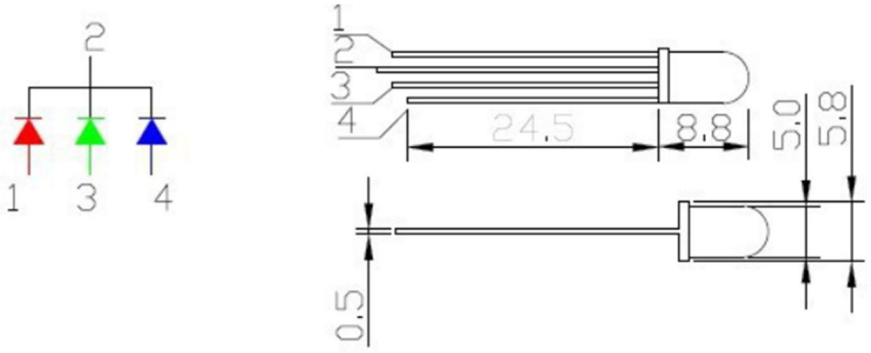

Рисунок 2.16 – Позначення виводів RGB-світлодіода та його розміри

Умовне графічне позначення RGB-світлодіода на електричній принциповій схемі зображено на рис. 2.17.

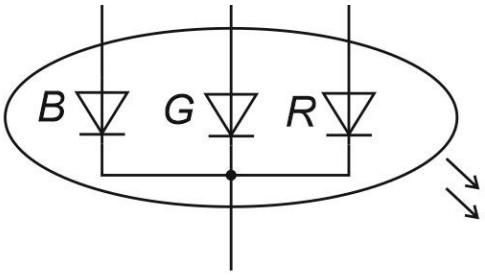

Рисунок 2.17 – Умовне графічне позначення RGB світлодіода на електричній принциповій схемі

Підключення виводів RGB-світлодіода до виводів ESP32 наведено в таблиці 2.5.

Таблиця 2.5 – Підключення виводів RGB-світлодіода до виводів ESP32

Вивід RGB-світлодіода	Вивід ESP32
Спільний катод	Будь-який вивід GND
R (Red)	D12
G (Green)	D14
B (Blue)	Не використовується

2.3 Опис електричної принципової схеми системи

Електричну принципову схему комп’ютерної системи табелювання робочого часу на основі технології RFID зображено на рис. 2.18.

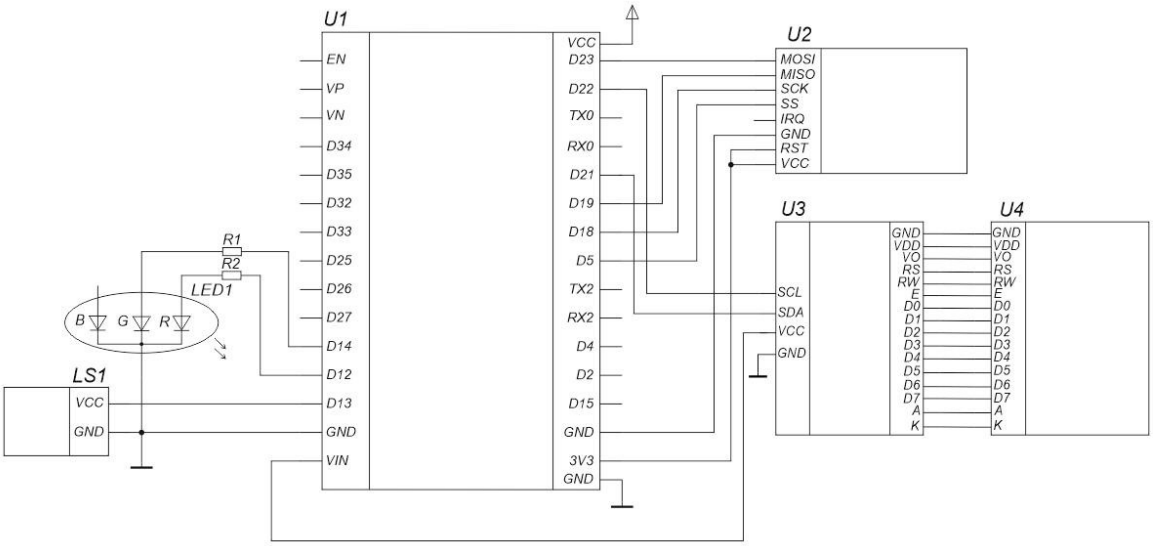

Рисунок 2.18 – Електрична принципова схема системи табелювання робочого часу на основі технології RFID

Позначення компонентів системи на електричній принциповій схемі:

– U1 – мікроконтролер ESP32;

– U2 – RFID-модуль RC522;

– U3 – I2C-конвертер;

– U4 – LCD 1602;

– LED1 – RGB-світлодіод марки SKV149;

– R1, R2 – резистори номіналом 200 Ом, 0,125 Вт;

– LS1 – активний п’єзодинамік.

2.4 Обґрунтування вибору програмного забезпечення для комп’ютерної системи табелювання робочого часу на основі технології RFID

2.4.1 Arduino IDE

Arduino IDE — це інтегроване середовище розробки від Arduino, програмне забезпечення з відкритим кодом, яке використовується для написання та завантаження програм (скетчів) на плати Arduino або плати інших розробників. Arduino IDE є багатоплатформовим додатком і працює на різних операційних системах. Включає в себе текстовий редактор для написання коду, компілятор та модуль завантаження скетчу в плату [13]. Головне вікно Arduino IDE зображено на рис. 2.19.

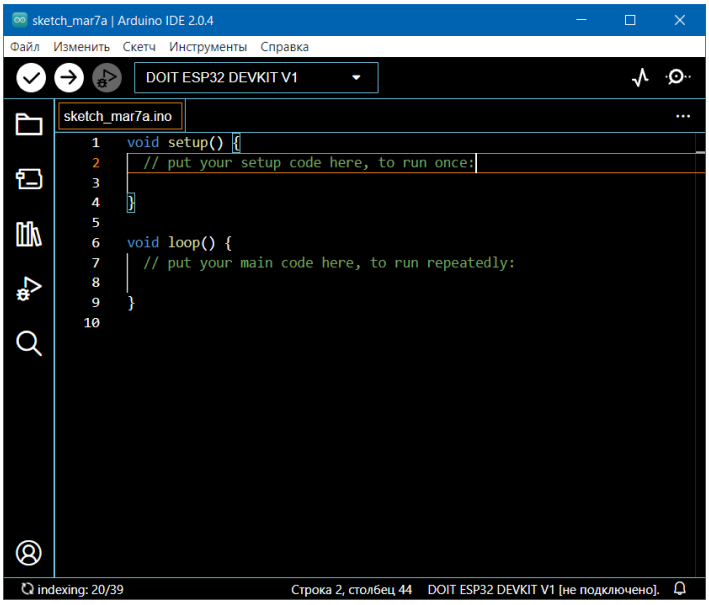

Рисунок 2.19 – Головне вікно Arduino IDE

Мова програмування, яку підтримує середовище розробки – Wiring. Фактично це мова програмування C/C++, але при цьому у ній додано декілька додаткових бібліотек.

2.4.2 IntelliJ IDEA

Розробку серверної частини системи, до якої відносяться розробка сайту та запуск сервера, реалізовано з допомогою інтегрованого середовища розробки IntelliJ IDEA [14].

IntelliJ IDEA (див. рис. 2.20) – це інтегроване середовище розробки програмного забезпечення для багатьох мов програмування.

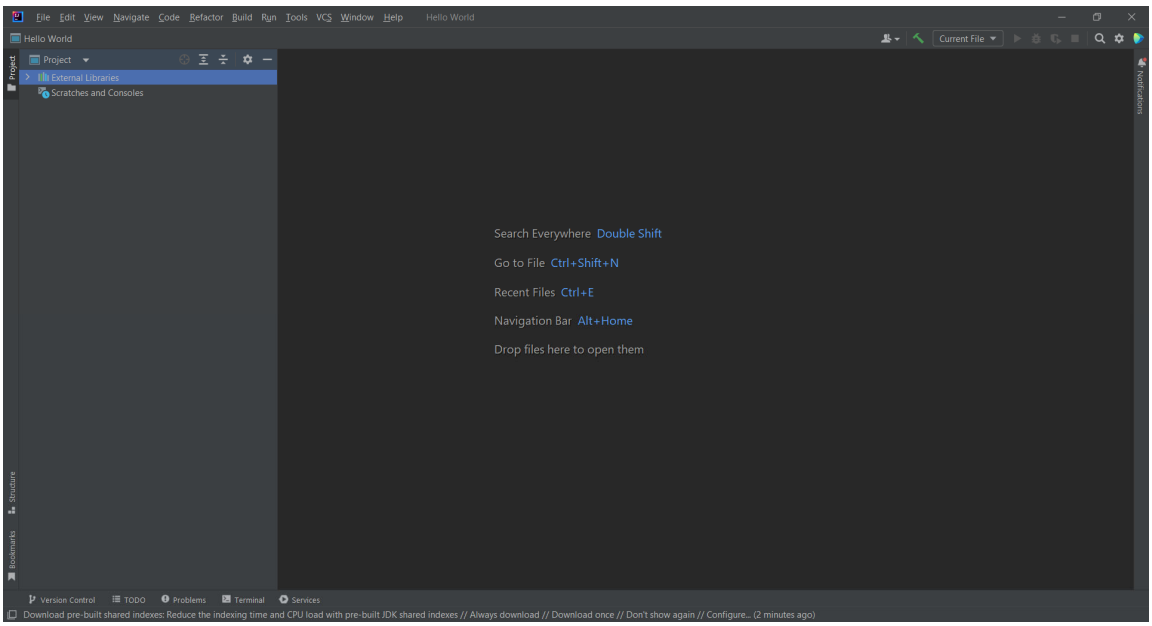

Рисунок 2.20 – Головне вікно проєкту IntelliJ IDEA

Дизайн інтегрованого середовища розробки орієнтований на продуктивність роботи, дозволяючи програмістам сконцентруватися на функціональних завданнях. IntelliJ IDEA при цьому бере на себе виконання буденних операцій.

Для написання серверної частини використано мову програмування Java, а для розмітки документа і стилізації – HTML5 та CSS3 відповідно.

2.4.3 Система керування базами даних MySQL

В якості системи керування базами даних (далі СКБД) використано MySQL.

MySQL – це вільна СКБД реляційного типу. Дана СКБД являється чудовим рішенням для малих та середніх програм.

Реляційна база даних – це тип бази даних, яка зберігає і надає доступ до точок даних, що пов’язані одина з одною. Реляційні бази даних базуються на

реляційній моделі – інтуїтивно зрозумілому та простому методі представлення даних у таблицях.

Рядки у таблиці являють собою набір пов’язаних значень. Кожен з них повинен бути позначений унікальним ідентифікатором, який називається Primary key (первинний ключ). Рядки таблиць також можуть бути пов’язані з рядками інших таблиць з допомогою зовнішніх ключів. Доступ до даних можна легко отримувати доступ багатьма способами. При цьому реорганізовувати таблиці у базі даних не потрібно.

Структуру таблиць бази даних, що використовується проєктованою системою, наведено в додатку Б.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Розробка алгоритму роботи комп'ютерної системи табелювання робочого часу на основі технології RFID

Алгоритм роботи комп'ютерної системи:

1. Ініціалізація бібліотек та оголошення глобальних змінних.

2. Процедура початкового запуску контролера:

– налаштування параметрів апаратного UART;

– ініціалізація модулів;

– налаштування виводів;

– підключення до точки доступу та перевірка роботи модуля Wi-Fi. Якщо підключення відсутнє, вивести відповідне повідомлення на LCD;

– зчитування точного часу та дати з NTP-сервера.

3. Вивести на дисплей системний час та дату, засвітити RGB-світлодіод червоним кольором.

4. Очікувати даних від RFID-модуля.

5. Якщо RFID-модуль зчитав дані з RFID-мітки та надіслав на зчитування, подати відповідний звуковий сигнал, зчитати UID мітки, засвітити RGB-світлодіод зеленим кольором.

6. Перевірити з'єднання з точкою доступу.

7. Якщо модуль Wi-Fi досі з'єднаний з точкою доступу, сформувати HTTP GET-запит з UID зчитаної мітки та надіслати його на сервер [10]. В іншому випадку вивести на монітор послідовного порта відповідну помилку.

8. Повернутися до очікування даних від RFID-модуля.

Блок-схема алгоритму роботи комп'ютерної системи табелювання робочого часу на основі технології RFID зображено на рис. 3.1-3.2.

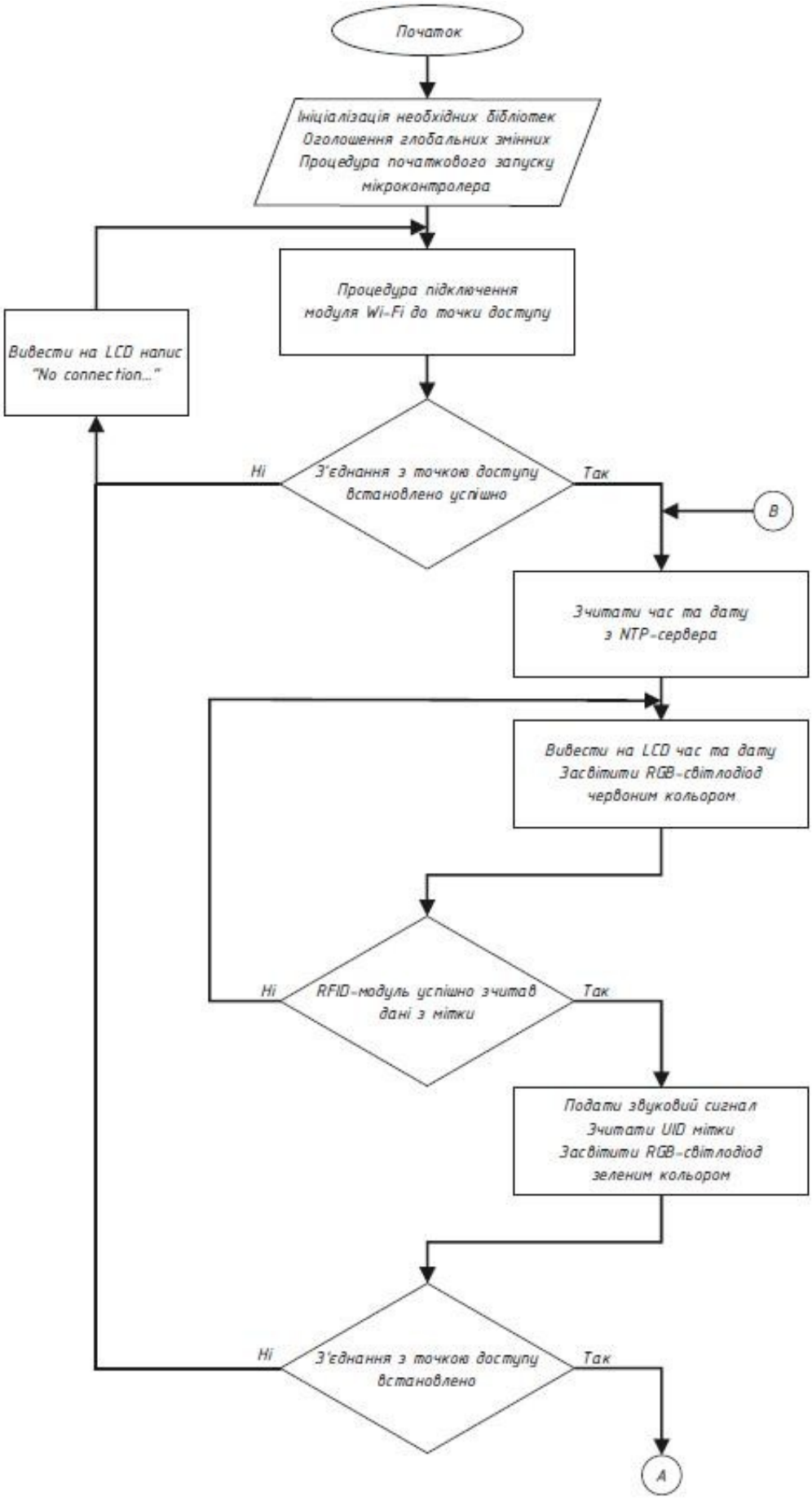

Рисунок 3.1 – Блок-схема алгоритму роботи комп'ютерної системи

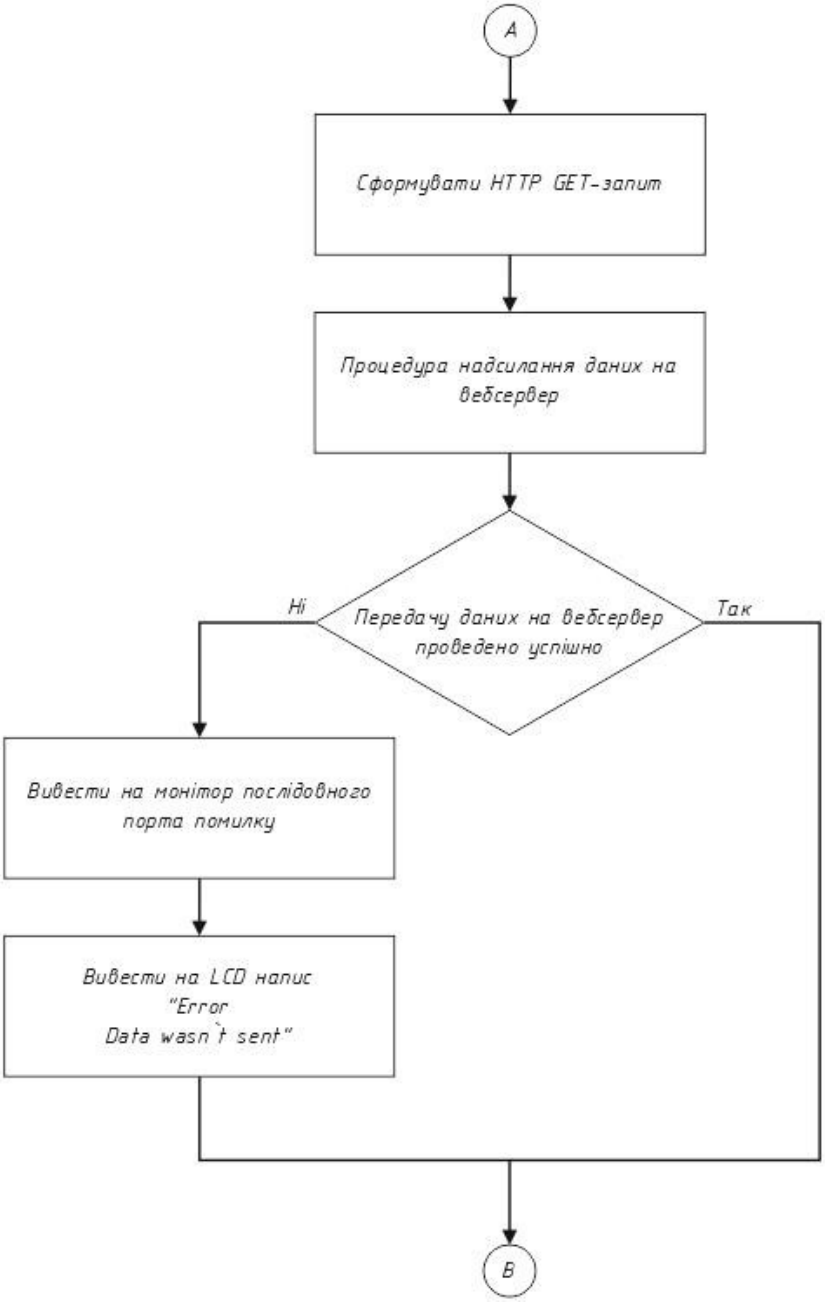

Рисунок 3.2 – Блок-схема алгоритму роботи комп'ютерної системи (продовження)

3.2 Написання кодів програми мікроконтролера ESP32

3.2.1 Ініціалізація бібліотек, оголошення глобальних змінних і об'єктів

Ініціалізацію бібліотек, оголошення глобальних змінних та об'єктів класів зображено на рис. 3.3.

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <SPI.h>
#include <MFRC522.h>
#include "time.h"
#include "LiquidCrystal_I2C.h"

#define RedRGB_Pin 12
#define GreenRGB_Pin 14
#define Buzzer_Pin 13
#define SS_Pin 5
#define RST_Pin 15

const char* Ssid = "MikroTik-DD086E";
const char* Password = "";
const char* NtpServer = "pool.ntp.org";
String serverName = "http://192.168.88.23:8080/controller?uid=";

LiquidCrystal_I2C lcd(0x27, 16, 2);
MFRC522 mfrc522(SS_Pin, RST_Pin);

hw_timer_t* timer = NULL;
volatile SemaphoreHandle_t timerSemaphore;
```

Рисунок 3.3 – Лістинг коду ініціалізації бібліотек, оголошення глобальних змінних та об’єктів класів

Бібліотеки, які використовуються в програмі:

– WiFi.h – призначена для керування та використання вбудованого у мікороконтролері ESP32 модуля Wi-Fi;

– HTTPClient.h – призначена для встановлення клієнт-серверного з’єднання, в якому ESP32 виступає клієнтом.;

– SPI.h – необхідна для взаємодії мікроконтролера ESP32 з RFID-модулем, який використовує SPI-протокол;

– MFRC522.h – призначена для керування RFID-модулем RC522;

– time.h – використовується для ініціалізації та керування локальним часом мікроконтролера ESP32;

– LiquidCrystal_I2C.h – бібліотека для керування LCD з допомогою інтерфейсу I2C.

Опис виводів та констант:

– RedRGB_Pin (D12) – вивід, до якого підключено ніжку «R» RGB-світлодіода;

– GreenRGB_Pin (D14) – вивід, до якого підключено ніжку «G» RGB-світлодіода;

– Buzzer_Pin (D13) – вивід, до якого підключено п’єзодинамік;

– SS_Pin (D5) – вивід, до якого підключено SS-вивід модуля RC522;

– RST_Pin (D15) – вивід, до якого підключено RST-вивід модуля RC522;

– Ssid – константа, яка містить у собі ім’я точки доступу Wi-Fi;

– Password – константа, яка містить пароль для з’єднання до точки доступу Wi-Fi;

– NtpServer – константа, що містить адресу NTP-сервера, який необхідний для зчитування точного часу та дати;

– ServerName – стрічка, що містить URL віддаленого сервера.

Також у лістингу наведено ініціалізацію об’єктів lcd (екземпляр класу LiquidCrystal_I2C), mfrc522 (екземпляр класу MFRC522), *timer (екземпляр класу hw_timer_t) та timerSemaphore (екземпляр класу SemaphoreHandle_t).

3.2.2 Процедура початкового запуску мікроконтролера

Процедура початкового запуску мікроконтролера (функція «setup()») виконується тільки один раз після запуску мікроконтролера ESP32 або його перезавантаження. Дана процедура використовується для встановлення режимів роботи необхідних виводів мікроконтролера, ініціалізації та запуску роботи послідовного асинхронного порта, ініціалізації роботи LCD, модуля RC522, шини SPI, з’єднання до точки доступу Wi-Fi, встановлення локального часу мікроконтролера та запуску таймера.

Лістинг функції «setup()» зображено на рис. 3.4.

```
void setup() {
  pinMode(RedRGB_Pin, OUTPUT);
  pinMode(GreenRGB_Pin, OUTPUT);
  pinMode(Buzzer_Pin, OUTPUT);
  digitalWrite(RedRGB_Pin, LOW);
  digitalWrite(GreenRGB_Pin, LOW);

  Serial.begin(115200);

  lcd.init();
  lcd.backlight();

  SPI.begin();
  mfrc522.PCD_Init();
  delay(4);

  wifiConnect();

  initTime("EET-2EEST,M3.5.0/3,M10.5.0/4");

  timerSemaphore = xSemaphoreCreateBinary();
  timer = timerBegin(0, 80, true);
  timerAttachInterrupt(timer, &onTimer, true);
  timerAlarmWrite(timer, 1000000, true);
  timerAlarmEnable(timer);
}
```

Рисунок 3.4 – Лістинг функції «setup()»

Таймер спрацьовує та виконує функцію переривання кожної 1 секунди.

Для підключення мікроконтролера ESP32 до точки доступу Wi-Fi використовується функція «wifiConnect()». Лістинг функції «wifiConnect()» зображено на рис. 3.5.

```
void wifiConnect() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("No connection...");
  WiFi.begin(Ssid, Password);
  Serial.println(F("Connecting"));
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print(F("Connected to WiFi network with IP Address: "));
  Serial.println(WiFi.localIP());
}
```

Рисунок 3.5 – Лістинг функції «wifiConnect()»

Для встановлення локального часу мікроконтролера використовується функція «initTime()», яка приймає один параметр – стрічку часового поясу. У даному проєкті використовується часовий пояс України (східноєвропейський стандартний час).

3.2.3 Процедура головного циклу

Процедура головного циклу (функція «loop()») – це процедура, яка виконується нескінченну кількість разів в ході роботи мікроконтролера.

У даній процедурі виконується оновлення дати та часу на LCD, зчитування даних з RFID-мітки, світлова та звукова індикація зчитування, а також надсилання даних на сервер з допомогою функції «sendGET()». Функція «sendGET()» приймає в якості параметру символьну стрічку.

Якщо жодної RFID-мітки не було зчитано, виконання функції «loop()» почнеться спочатку.

Лістинг зчитування UID RFID-мітки зображено на рис. 3.6.

```
String strUID = "";
Serial.print(F("Card UID: "));
for (byte i = 0; i < mfrc522.uid.size; i++) {
  Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
  Serial.print(mfrc522.uid.uidByte[i], HEX);
  strUID.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
  strUID.concat(String(mfrc522.uid.uidByte[i], HEX));
}
Serial.println();
```

Рисунок 3.6 – Лістинг зчитування UID RFID-мітки

Функція «sendGET()» використовується для встановлення HTTP-з’єднання з сервером та надсилання йому GET-запиту, в якому міститься зчитаний UID RFID-мітки.

3.3 Створення та налаштування проєкту в IntelliJ IDEA

Для створення проєкту обрано один з архетипів Maven – maven-archetype-webapp. Maven – це набір засобів для автоматизації роботи з програмними проєктами, використовується для керування та складання (компонування) програм. Архетип Maven – це шаблонний проєкт.

Для розробки проєктів на мові Java також необхідний JDK – набір інструментів розробки Java-проєктів. У даному проєктуванні використано OpenJDK-19 [15].

Далі після створення проєкту потрібно налаштувати запуск локального сервера. Для цього у даному проєкті використовується контейнер сервлетів Apache Tomcat версії 9.0.73.

Apache Tomcat – це контейнер сервлетів з відкритим початковим кодом, який також виконує функцію вебсервера. Фактично контейнер сервлетів це Java-застосунок, який забезпечує взаємозв’язок між сервером та клієнтами [16].

Java Servlet API (сервлет) – це стандартизований прикладний програмний інтерфейс для створення динамічного контенту до вебсервера, використовуючи платформу Java.

Для налаштування проєкту спочатку завантажено архів Tomcat-9.0.73 з офіційного сайту Apache Tomcat, після чого розпаковано його. Далі на панелі інструментів вибрано випадаюче меню “Current File” та “Edit Configurations”. Після цього відкривається вікно “Run/Debug Configurations”.

Далі необхідно додати нову конфігурацію. Для цього натиснуто на відповідну кнопку у вигляді знака “+” та серед всіх можливих конфігурацій вибрано Smart Tomcat. В якості значення “Tomcat server” вибрано шлях до папки з Apache Tomcat-9.0.73, яку було видобуто з архіву, завантаженого з офіційного сайту Apache Tomcat. Усі інші значення залишено за замовчуванням.

3.4 Написання кодів серверної частини

Проєкт створено на основі патерну об’єктно-орієнтованого програмування MVC.

Патерн MVC розшифровується як патерн Model-View-Controller. Він передбачає розподіл відповідних основних елементів програми, а саме:

- Model – являє собою об'єкт, який вміщує або переносить у собі певні дані. Він також може мати логіку оновлення контролера, якщо його дані змінюються.

- View – являє собою візуалізацію даних, які містить Model. Зазвичай це інтерфейс користувача.

- Controller – являє собою елемент, який обробляє дії користувача та контролює потік даних в Model та оновлює представлення (елемент View) щоразу, коли дані змінюються. При цьому він зберігає розподіл елементів Model та View [17].

Пакети, які використовуються у проєкті:

– java – вміщує у собі підкаталоги з java-файлами (з розширенням .java);

– ua.tntu.server – проміжні пакети;

– controller – вміщує у собі java-файл сервлета «контролер», який відповідає за керування даними;

– dao – data access object, каталог, що вміщує у собі необхідні java-файли абстрактних інтерфейсів для доступу до бази даних;

– model – вміщує у собі java-файли, які використовуються виключно в якості даних. В даному проєкті вони являють собою шаблонну структуру таблиць у базі даних;

– service – вміщує у собі java-файли сервісів, які забезпечують логіку роботи сервера;

– util – вміщує у собі java-файли класів «HibernateUtil» та «ScheduleCheck».

– resourses – вміщує у собі файл фреймворка «Hibernate»;

– webapp – вміщує у собі файли вебпрограм (API);

– WEB-INF – вміщує у собі файл web.xml.

3.4.1 Файли пакету «model»

У пакеті «model» створено 4 java-файли, а саме: «Employee», «Entry», «User» та «WorkDay». У кожному з цих файлів описано відповідні публічні класи, екземпляри яких являють собою таблиці, а їх параметри - назви колонок.

Об’єкти класу «Employee» вміщують у собі інформацію про працівника на підприємстві, його статус на роботі та розклад робочого дня.

Лістинг коду класу «Employee» та його параметрів зображено на рис. 3.7.

```
@Entity
@Table(name = "employeeEntity")
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "employeeId")
    private int employeeId;
    @Column(name = "employeeName")
    private String employeeName;
    @Column(name = "employeePosition")
    private String employeePosition;
    @Column(name = "employeeStatus")
    private String employeeStatus;
    @Column(name = "startTime")
    private LocalTime startTime;
    @Column(name = "endTime")
    private LocalTime endTime;
    @Column(name = "cardCode")
    private String cardCode;

    @OneToOne(mappedBy = "employee")
    private User user;
    @OneToMany(mappedBy = "employee")
    private List<Entry> entryList;
    @OneToMany(mappedBy = "employee")
    private List<WorkDay> dayList;
}
```

Рисунок 3.7 – Лістинг коду класу «Employee» та його параметрів

3.4.2 Файли пакету «dao»

DAO – це абстрактний інтерфейс до деяких типів баз даних або механізмів збереження даних, що реалізує певні операції без розкриття деталей бази даних.

У пакеті «dao» створено файли «EmployeeDAO», «EntryDAO», «UserDAO» та «WorkDayDAO», у яких реалізовано опис відповідних однойменних класів.

Усі класи з пакету «dao» містять методи з аналогічним функціоналом. До таких функцій відносяться:

– методи створення запису у таблиці: addEmployee(), addEntry(), addUser() та addWorkDay();

– методи оновлення запису у таблиці: updateEmployee(), updateEntry(), updateUser() та updateWorkDay();

– методи видалення запису у таблиці: deleteEmployee(), deleteEntry(), deleteUser() та deleteWorkDay();

– методи зчитування запису з таблиці: getEmployee(), getEntry(), getUser() та getWorkDay();

– методи зчитування усіх записів з таблиці: getAllEmployee(), getAllEntry(), getAllUser(), getAllWorkDay().

Лістинг коду методу «addEmployee()» наведено на рис. 3.8.

```
public void addEmployee(Employee employee){
    Transaction transaction = null;
    try(Session session = HibernateUtil.getSessionFactory().openSession()){
        transaction = session.beginTransaction();
        session.save(employee);
        transaction.commit();
    } catch (Exception e) {
        if (transaction != null) {
            transaction.rollback();
        }
        e.printStackTrace();
    }
}
```

Рисунок 3.8 – Лістинг коду методу «addEmployee()»

Робота кожного з методів класів DAO забезпечується на основі транзакцій.

Транзакція бази даних – це послідовність кількох операція, які виконуються над базою даних. При цьому дані операції виконуються як єдина логічна обиниця роботи, тобто або виконуються повністю, або ж не виконуються взагалі. Виконуючи транзакцію, ніколи не буває ситуації, щоб було виконано лише половину операцій і збережено їх результати виконання.

3.4.3 Файли пакету «service»

У пакеті «service» створено 4 java-файли з назвами «EmployeeService», «EntryService», «UserService» та «WorkDayService». У кожному з цих файлів описано відповідні однойменні класи.

Об’єкти класів з пакету «service» використовуються для доступу до виконання методів, які описано у відповідних файлах пакету «dao».

Лістинг коду опису класу «EmployeeService» зображено на рис. 3.9.

```
public class EmployeeService {
    EmployeeDAO employeeDAO;

    public EmployeeService(EmployeeDAO employeeDAO) {
        this.employeeDAO = employeeDAO;
    }

    public void addEmployee(Employee employee){
        employeeDAO.addEmployee(employee);
    }
    public void updateEmployee(Employee employee){
        employeeDAO.updateEmployee(employee);
    }

    public void deleteEmployee(int id){
        employeeDAO.deleteEmployee(id);
    }
    public Employee getEmployee(int id) {
        return employeeDAO.getEmployee(id);
    }
    public Employee getEmployeeByCode(String code) {
        return employeeDAO.getEmployeeByCode(code);
    }
    public List<Employee> getAllEmployee() {
        return employeeDAO.getAllEmployee();
    }
}
```

Рисунок 3.9 – Лістинг коду опису класу «EmployeeService»

3.4.4 Файли пакету «util»

У пакеті «util» створено два java-файли, у яких описано класи «HibernateUtil» та «ScheduleCheck».

Hibernate – це бібліотека для мови програмування Java, що призначена для вирішення задач об’єктно-реляційного відображення (ORM) [18]. При ініціалізації об’єкту класу «HibernateUtil» ініціалізується незмінний потокобезпечний об’єкт класу «SessionFactory» з компільованим мапінгом для однієї бази даних.

3.4.5 Java-файл «WebController»

Java-файл «WebController» розташований у пакеті «controller». У даному файлі описано клас «WebController», що наслідується від абстрактного класу «HttpServlet», що фактично робить його сервлетом.

Клас «WebController» містить у собі 2 параметри та 4 методи.

Методи класу «WebController»:

– init() – використовується для ініціалізації таймера для виконання регулярного методу «run()» класу «ScheduleCheck»;

– doGet() – перевизначений метод абстрактного класу «HttpServlet». Використовується для перенаправлення запитів. Виконує перенаправлення метод «getRequestDispatcher()»;

– doPost() – перевизначений метод абстрактного класу «HttpServlet». Використовується для переадресації запитів. Виконує переадресацію метод «sendRedirect()»;

– process() – основний метод сервлета «WebController». У даному методі обробляються запити від користувачів та формуються відповіді на дані запити.

Лістинг коду обробки UID, надісланого мікроконтролером ESP32, та встановлення відповідному працівнику поточного статусу зображено на рис. 3.10.

```
if(commandName.equals("espData")){
    String code = request.getParameter("uid");
    Employee employee = employeeService.getEmployeeByCode(code);
    employee.setEmployeeStatus(employee.getEmployeeStatus().equals("Active") ? "Absent" : "Active");
    entryService.addEntry(new Entry(
            employee,
            LocalDateTime.now(),
            employee.getEmployeeStatus()
    ));
}
```

Рисунок 3.10 – Лістинг коду обробки UID та встановлення відповідному працівнику поточного статусу

В залежності від ролі користувача («Admin» або «User») йому повертаються різні вебсторінки. Користувачу з роллю «Admin» повертається сторінка «main.jsp», а користувачу з роллю «User» - «employeeInfo.jsp».

3.4.6 JSP та CSS файли

Файл JSP – це вебсторінка, що включає у собі html-код та програмний код на мові програмування Java. Коли клієнт подає запит до конкретної сторінки JSP, то сервер спочатку обробляє дану сторінку, генерує із неї html-код і повертає його клієнту. В результаті клієнт бачить звичайну html-сторінку.

У контейнері «webapp» створено 5 вебсторінок JSP:

– logIn.jsp – вебсторінка, призначена для аутентифікації користувача або адміністратора;

– signUp.jsp – вебсторінка, призначена для створення нового користувача;

– main.jsp – вебсторінка, доступ до якої надається тільки адміністратору. Призначена для відображення списку працівників та їх статусу у вигляді таблиці;

– employeeInfo.jsp – вебсторінка, призначена для відображення інформації про працівника;

– accountInfo.jsp – вебсторінка, призначена для відображення інформації про користувача.

Для спрощення вбудовування java-коду в JSP використовується бібліотека «JSTL».

Для стилізації вебсторінок використовуються файли CSS, які підключаються до файлів JSP. Всього створено 3 файли CSS:

– «logInStyles.css» – вміщує у собі стилі для вебсторінок «logIn.jsp» та «signUp.jsp»;

– «navStyles» - вміщує у собі стилі для навігаційної панелі на вебсторінках та анімації для них;

– «contentStyles» - вміщує у собі стилі для головного контенту на вебсторінках.

3.5 Результати роботи системи

Дослідний взірець апаратної частини системи табелювання робочого часу на основі RFID-технології зображено на рис. 3.11. Компоненти апаратної частини проєктованої системи з’єднано між собою відповідно до електричної принципової схеми (див. рис. 2.18) із використанням провідників та макетної плати.

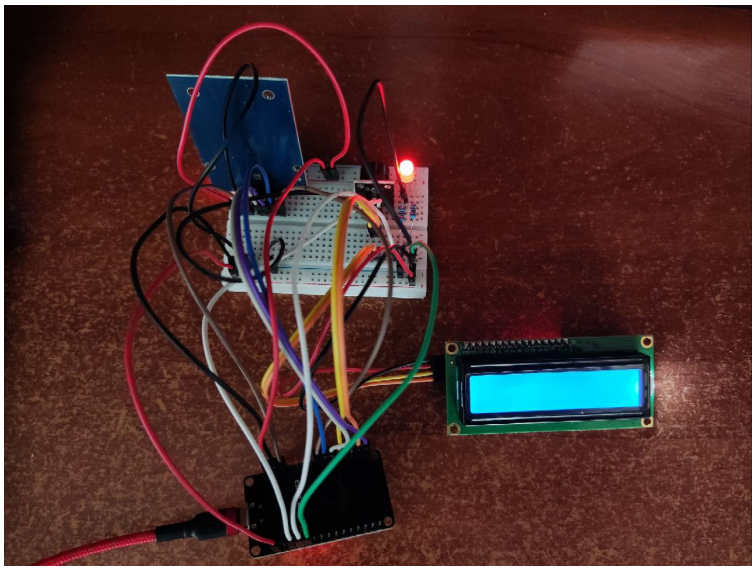

Рисунок 3.11 – Дослідний взірець апаратної частини системи табелювання робочого часу на основі RFID-технології

Після подачі живлення пристрій запускається, після чого виконується ініціалізація усіх компонентів та підключення мікроконтролера до точки доступу Wi-Fi. При даній процедурі RGB-світлодіод засвічується жовтим кольором, а на LCD відображається напис «No connection…».

Якщо мікроконтролер підключився до точки доступу Wi-Fi та йому вдалось з'єднатися з NTP-сервером, на LCD відображаються точні дата та час (рис. 3.12).

Рисунок 3.12 – Відображення точної дати та часу на LCD

Після того, як на LCD відображено точні дата та час, мікроконтролер у стані очікування зчитування карти доступу. Якщо карту доступу зчитано RFID-модулем, в момент зчитування пролунає звуковий сигнал, RGB-світлодіод засвітиться зеленим кольором, а мікроконтролер розпочне передачу зчитаних даних на сервер. При передачі даних на сервер на LCD відображається напис «Sending data…».

Якщо при надсиланні даних виникла якась помилка, на LCD виводиться напис «Error: %d Data wasn`t sent», де %d – це код помилки.

Якщо дані надіслано успішно, світлодіод засвітиться червоним кольором, а на LCD знову відобразиться точні дата та час, мікроконтролер повернеться у стан очікування зчитування карти доступу.

Для перевірки працездатності серверної частини проведено тестування функціональності вебсайту та його вебсторінок.

Тестування сторінки аутентифікації користувача зображено на рис. 3.13.

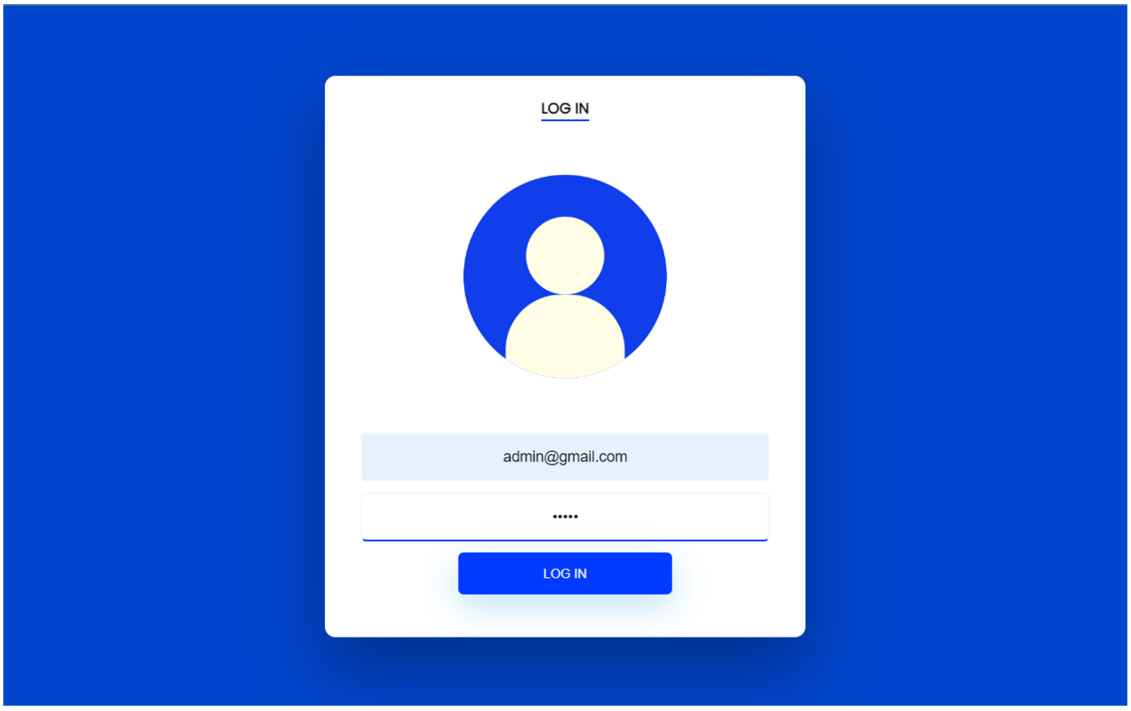

Рисунок 3.13 – Тестування сторінки аутентифікації користувача

Тестування сторінки «signUp.jsp» зображено на рис. 3.14.

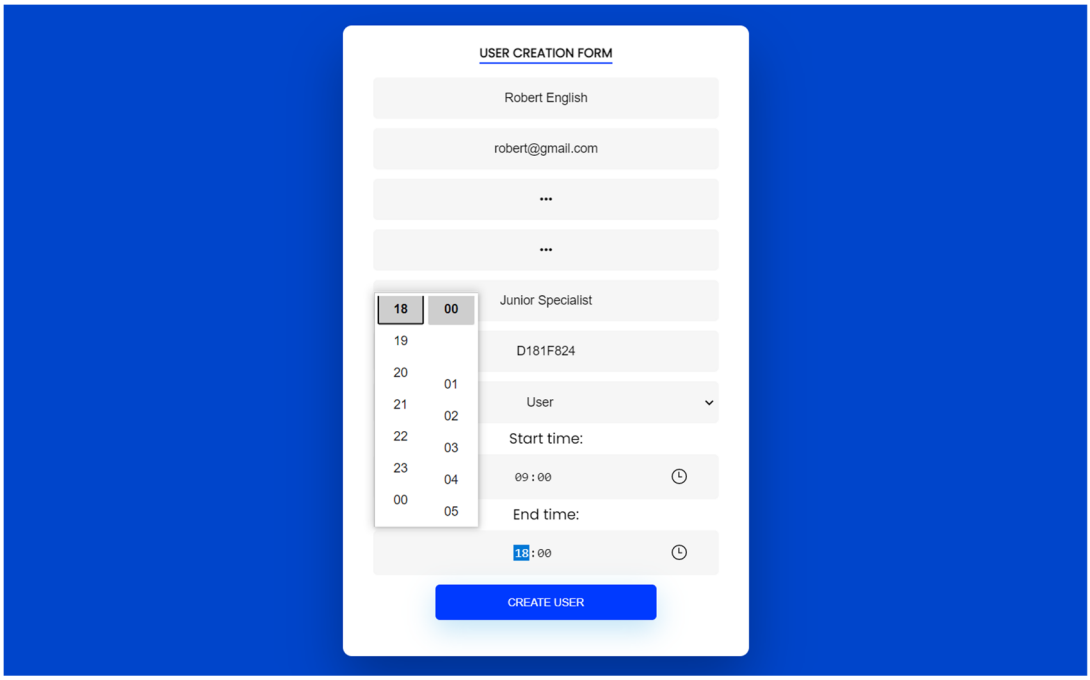

Рисунок 3.14 – Тестування сторінки «signUp.jsp»

Тестування сторінки «employeeInfo.jsp» зображено на рис. 3.15.

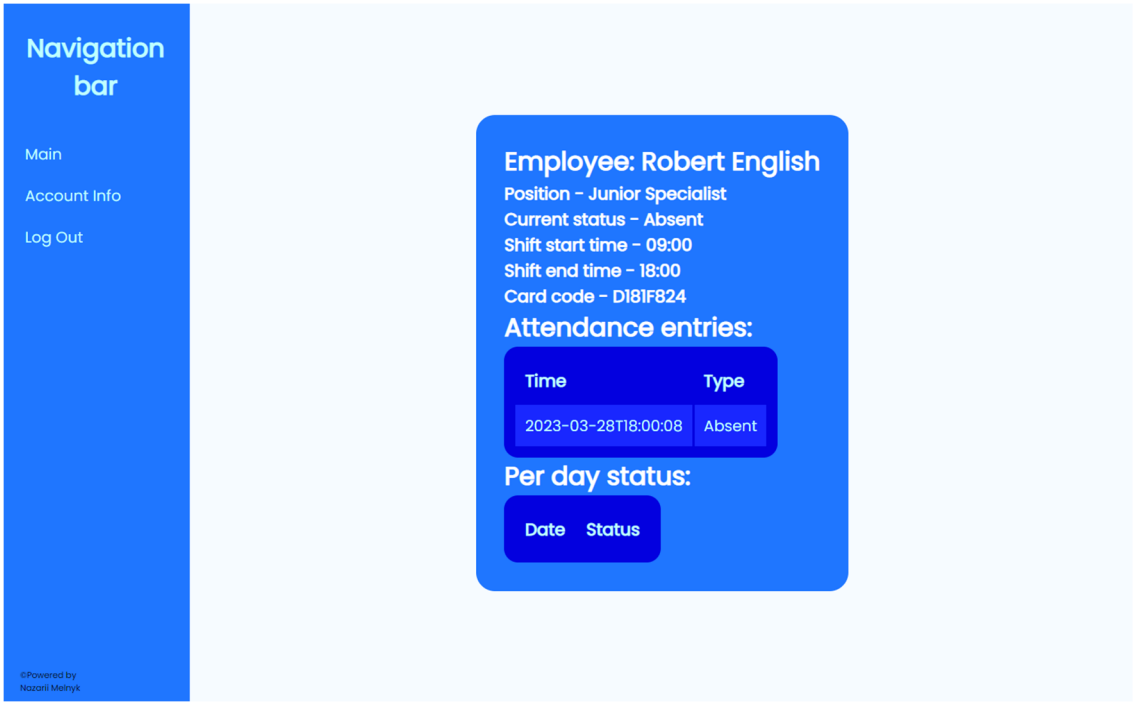

Рисунок 3.15 – Тестування сторінки «employeeInfo.jsp»

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

Темою кваліфікаційної роботи бакалавра є розробка комп'ютерної системи табелювання робочого часу на основі технології RFID. Відповідно з безпеки життєдіяльності та основ охорони праці розглянуто питання:

– працездатність оператора системи при моніторингу і табелюванні робочого часу;

– вимоги пожежної безпеки при гасінні електроустановок.

4.1 Працездатність оператора системи при моніторингу і табелюванні робочого часу

Працездатність людини – це можливість виконувати роботу з необхідною якістю в установлений час. Працездатність людини залежить від зовнішніх та внутрішніх чинників.

До зовнішніх належать:

– кількість та форма отриманої інформації;

– зручність робочого місця;

– характер взаємовідносин у колективі.

До внутрішніх належать:

– рівень підготовки;

– тренованість людини;

– емоційна стійкість.

Під час роботи людина переживає різні функціональні стани, які зумовлюють різні рівні її працездатності. Зміни функціонального стану та якості роботи людини під час однієї зміни показано на рис. 4.1.

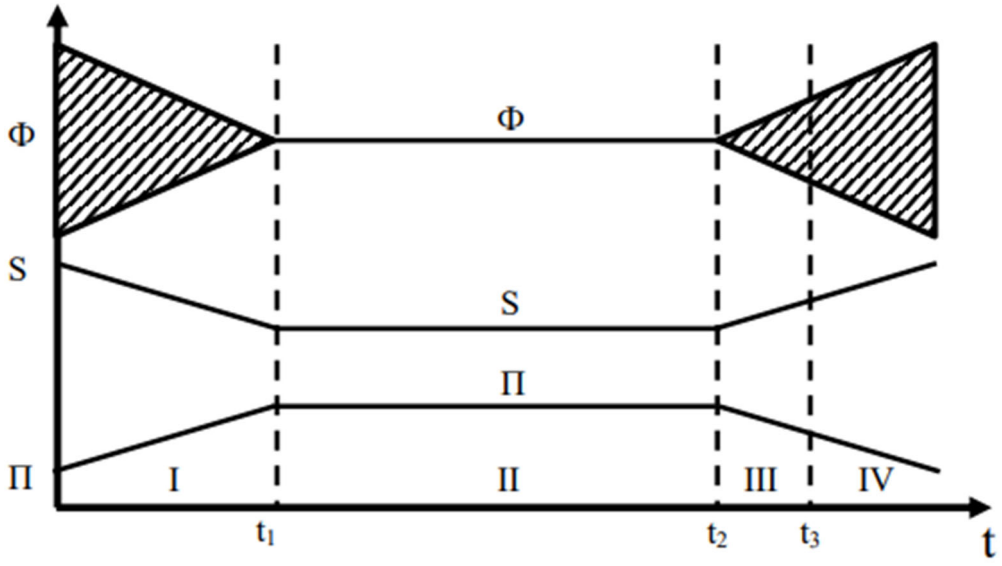

Рисунок 4.1 – Фази працездатності

Позначення на графіку:

– Ф – показник функціонального стану;

– S – помилки в роботі;

– П – продуктивність.

Фаза пристосування до праці (період I) – це час, протягом якого людина адаптується до умов праці. Показник поступово досягає певного встановленого значення. Тривалість періоду пристосування організму до умов праці залежить від багатьох чинників. Основними серед цих чинників є інтенсивність роботи та рівень готовності людини до роботи. Значного пришвидшення фази адаптації до праці можна досягти завдяки попередній підготовці людини до роботи, а також посиленому розумовому навантаженню.

Фаза стійкої працездатності (період II) характеризується найвищою якістю праці при оптимальному рівні функціонування фізіологічних систем людського організму. Тривалість залежить від інтенсивності роботи, тобто чим інтенсивніша праця, тим коротший даний період.

Продовження періоду стійкої працездатності забезпечується:

– оптимальним рівнем напруги психофізіологічних функцій;

– комфортними умовами праці;

– правильним поєднанням режимів праці та відпочинку;

– емоційним розвантаженням;

– використанням тонізуючих напоїв та фармакологічних засобів, зокрема препаратів рослинного походження;

– інформуванням людини про наслідки її діяльності, наглядом та контролем її роботи.

Фаза субкомпенсації (період III) – початок розвитку втоми. У цей період якість праці ще зберігається на високому рівні, проте це забезпечується тільки завдяки перенапрузі та відповідним функціям людського організму.

Фаза втоми (період IV) – це фаза, яка характеризується чітко вираженим зниженням в якості роботи при подальших погіршеннях функціонального стану людини. Об’єктивними показниками втоми є зміна частоти пульсу, дихання, зорової та слухової чутливості.

Після виконання своєї роботи наступає фаза відновлення. Під час даної фази працівник відпочиває і вона може тривати як декілька хвилин / годин, так і навіть декілька діб [19].

При роботі оператора системи табелювання робочого часу на основі RFID-технології враховуються чотири фази працездатності, а також період відновлення. Правильний розрахунок часу, відведеного для кожної фази працездатності, а також достатній період відновлення після роботи запобігає погіршенню фізичного та ментального здоров’я працівника, при цьому забезпечуючи більшу стійкість працездатності та стабільність в роботі.

4.2 Вимоги пожежної безпеки при гасінні електроустановок

Пожежна безпека у підприємств та організацій, які використовують електроустановки, забезпечується шляхом здійснення організаційно-технічних, а також інших заходів з попередження виникнення різних пожеж,

зменшення можливих матеріальних збитків, забезпечення безпеки, запобіганню великих негативних екологічних наслідків, створення умов для успішного пожежогасіння та швидкого виклику пожежних.

Апаратна частина комп’ютерної системи табелювання робочого часу передбачає встановлення блоку живлення для забезпечення необхідної напруги, а такі електроустановки при неналежному нагляді можуть спалахнути та загорітися.

Існує два методи для гасіння електроустановок, які використовуються при виникненні пожежі:

– гасіння електроустановок відділених від напруги мережі;

– гасіння електроустановок, які знаходяться під напругою.

Гасіння обладнання, що працює під напругою, вимагає застосування спеціальних засобів та дотримання спеціальних заходів з техніки безпеки.

Необхідно враховувати наступні складнощі:

– при гасінні необхідно враховувати особливості обладнання, наявність легкозаймистих речовин, які можуть стати джерелом додаткового займання;

– при гасінні слід виключити ситуації, у яких займання може перекинутися на розподільні ділянки, що знаходяться поруч;

– необхідно вжити заходів для захисту від розплавлення ізоляції, підвищення температури та появи додаткового джерела задимлення, займання;

– при ліквідації пожеж із загорянням водню застосовуються речовини, що його витісняють, тобто усувається джерело поширення полум'я.

Усі електроустановки перед гасінням слід знеструмити, відключивши від джерел живлення, та заземлити. За можливістю навколо влаштовується теплоізоляція для захисту решти обладнання або вживаються заходи для запобігання переходу полум'я на сусідні установки. Гасіння здійснюється від краю ділянки, просуваючись усередину та виключаючи ймовірність поширення полум'я [19].

Для забезпечення безпеки персоналу та пожежників, які беруть участь у гасінні пожежі електроустановок під напругою, застосовуються індивідуальні ізолюючі електрозахисні засоби (діелектричні рукавиці, калоші, килими).

При виникненні пожежі на електроустановці, особа, яка першою виявила пожежу, для запобігання подальшого розгорання повинна негайно повідомити про неї.

Гасіння електроустановок під напругою з використанням ручних стволів повинне здійснюватися за таких умов:

– застосування ефективних способів спеціальних речовин в зону горіння;

– дотримання безпечних відстаней до електроустановок, які знаходяться під напругою;

– застосування індивідуальних ізолюючих електрозахисних засобів при гасінні пожеж електроустановок без зняття напруги;

– забезпечення надійного заземлення стволів і пожежних автомобілів.

В якості речовин для гасіння електроустановок під напругою необхідно використовувати:

– розпилені струмені води;

– інертні гази й порошкові суміші;

– комбіновані суміші.

Використовувати усі види піни при гасінні таких пожеж заборонено.

Під час гасіння пожежі на електроустановках під напругою необхідно застосовувати засоби та прийоми подачі в зону горіння вогнегасних речовин, які забезпечують безпечну роботу пожежників і ефективне гасіння пожежі.

Під час пожежі на електроустановках під напругою обслуговуючий персонал зобов'язаний у першу чергу повідомити про пожежу начальника зміни (чергового, диспетчера) й пожежну охорону, а потім ужити всіх необхідних заходів відповідно до плану пожежогасіння [19].

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи бакалавра вирішено актуальне технічне завдання, яке полягає у розробці комп'ютерної системи табелювання робочого часу на основі технології RFID.

В процесі дипломного проєктування отримано такі практичні результати:

– розглянуто та проаналізовано сучасні системи для табелювання робочого часу на підприємствах;

– розроблено структурну схему системи табелювання робочого часу на основі технології RFID;

– здійснено вибір необхідних компонентів для реалізації проєктованої системи;

– розроблено електричну принципову схему комп'ютерної системи, використовуючи вибрані електронні елементи;

– написано алгоритм функціонування системи, а також розроблено відповідне алгоритму програмне забезпечення для мікроконтролера;

– розроблено вебсайт та написано програмне забезпечення для серверної частини комп'ютерної системи.

В результаті виконання кваліфікаційної роботи отримано робочий прототип комп'ютерної системи табелювання робочого часу на основі технології RFID, який характеризується високою продуктивністю та низькою вартістю.

Тестування комп'ютерної системи підтвердило її працездатність. Розроблена система придатна до застосування в реальних умовах експлуатації на підприємтсві для табелювання робочого часу працівників.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Осухівська Г.М., Тиш Є.В., Луцик Н.С., Паламар А.М. Методичні вказівки до виконання кваліфікаційних робіт здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 123 «Комп'ютерна інженерія» усіх форм навчання. Тернопіль, ТНТУ. 2022. 28 с.

2. Табель обліку робочого часу. URL: https://biz.ligazakon.net/aktualno/7756_tabel-oblku-robochogo-chasu (дата звернення: 22.01.2023 р.)

3. Порядок ведення табеля обліку робочого часу. URL: https://ips.ligazakon.net/document/reader/BZ013201 (дата звернення: 22.01.2023р.)

4. Термінал контролю доступу та обліку робочого часу Dahua DHI-ASA1222G. URL: https://dahua.company/product/DHI-ASA1222G (дата звернення: 10.02.2023 р.)

5. Програмний продукт трекінгу робочого часу Time Analytics. URL: https://timeanalyticssoftware.com (дата звернення: 10.02.2023 р.)

6. Офіційний сайт Espressif. URL: https://www.espressif.com (дата звернення: 27.02.2023 р.)

7. Інформація про мікроконтролер Espressif ESP32. URL: https://www.espressif.com/en/products/socs/esp32 (дата звернення: 27.02.2023р.)

8. Технічна документація до мікроконтролера Espressif ESP32. URL: https://www.espressif.com/en/support/documents/technical-documents (дата звернення: 27.02.2023 р.)

9. Технічна документація модуля MFRC522. URL: https://pdf1.alldatasheet.com/datasheet-pdf/view/346109/NXP/RC522.html (дата звернення: 08.03.2023 р.)

10. Технічна документація LCD 1602. URL: https://datasheetspdf.com/pdf-file/519148/CA/LCD-1602A/1 (дата звернення: 10.03.2023 р.)

11. Технічна документація модуля активного п’єзодинаміка. URL: https://datasheetspdf.com/datasheet/KY-012.html (дата звернення: 15.03.2023 р.)

12. Технічна документація RGB-світлодіода із загальним катодом. URL: https://arduino.ua/docs/LEDRGB5mm.pdf (дата звернення: 20.03.2023 р.)

13. Сторінка з інформацією про Arduino IDE. URL: https://www.arduino.cc/en/software (дата звернення: 06.04.2023 р.)

14. Офіційний сайт IntelliJ IDEA. URL: https://www.jetbrains.com/idea/ (дата звернення: 13.04.2023 р.)

15. Офіційний сайт OpenJDK. URL: https://openjdk.org (дата звернення: 15.04.2023 р.)

16. Офіційний сайт Apache Tomcat. URL: https://tomcat.apache.org (дата звернення: 10.05.2023 р.)

17. Інформація про патерн програмування MVC. URL: https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm (дата звернення: 12.05.2023 р.)

18. Документація Hibernate ORM. URL: https://hibernate.org/orm/documentation/6.2/ (дата звернення: 15.05.2023 р.)

19. Бедрій І.Я., Нечай В.Я. Безпека життєдіяльності. Навчальний посібник. – Львів: Манголія 2006, 2007. 499 с.

20. Tysh Ie. Approach and method of evaluation of the general reliability indicator of computer systems. International scientific journal “Computer systems and information technologies”, 3 (5). Khmelnytskyi National University. 2021. P.74-80.

21. Юськів Я., Тиш Є. База даних підтримки процесу оцінювання впливу дефектів програмного забезпечення на надійність комп’ютерних систем. Матеріали VII науково-технічної конференції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя, (Тернопіль, 11 – 12 грудня 2019 р.). 2019. С. 146.

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Кафедра комп’ютерних систем та мереж

«ЗАТВЕРДЖУЮ»

Завідувач кафедрою КС

_____ Осухівська Г.М.

“_____” _____ 2023 р.

КОМП’ЮТЕРНА СИСТЕМА ТАБЕЛЮВАННЯ РОБОЧОГО ЧАСУ НА ОСНОВІ RFID-ТЕХНОЛОГІЇ

ТЕХНІЧНЕ ЗАВДАННЯ

на ___8___ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп’ютерна інженерія»

«УЗГОДЖЕНО»	«ВИКОНАВЕЦЬ»
Керівник кваліфікаційної роботи	Студент групи СІс-41
_____ к.т.н. Тиш Є. В.	_____ Мельник Н. О.
“_____” _____ 2023 р.	“_____” _____ 2023 р.

Тернопіль 2023

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи бакалавра: «Комп’ютерна система табелювання робочого часу на основі RFID-технологїі».

Умовне позначення кваліфікаційної роботи бакалавра: КС КРБ 123.357.00.00

1.2 Виконавець

Студент групи СІс-41, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Мельник Назарій Олександрович.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи бакалавра є наказ по університету № *4/7-237* від «*28*» *лютого* 20*23* року.

1.4 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи бакалавра – 23.01.2023 р.

Плановий термін завершення виконання кваліфікаційної роботи бакалавра – 15.06.2023 р.

1.5 Порядок оформлення та пред’явлення результатів роботи

Оформлення технічної документації до кваліфікаційної роботи бакалавра здійснюється згідно діючих вимог вітчизняних та міжнародних стандартів. Технічна документація до кваліфікаційної роботи бакалавра включає в себе текст пояснювальної записки та креслення, які максимально інформативно та стисло відображають основні результати розробки комп’ютерної системи табелювання робочого часу на основі RFID-технології. Основними регламентними документами при оформленні та пред’явлені результатів проєктування є групи діючих стандартів ДСТУ, ГОСТ, ISO та ЄСКД, ЕСПД. Пред’явлення результатів кваліфікаційної роботи бакалавра відбувається шляхом захисту дипломного проєкту на відповідному засіданні ДЕК, ілюстрацією основних досягнень за допомогою графічного матеріалу.

2 Призначення і цілі створення системи

2.1 Призначення системи

Система призначена для табелювання робочого часу на підприємстві.

2.2 Мета створення системи

Метою створення системи є:

– зчитування інформації про працівника з допомогою RFID-технології;

– передача інформації про працівника та точний час зчитування даної інформації на віддалений сервер;

– відображення інформації на сайті про відвідуваність працівника на робочому місці.

2.3 Характеристика об’єкту

Система проєктується для табелювання робочого часу на підприємстві з допомогою RFID-технології, що включає в себе:

– розробку функціональної та структурної схем;

– розробку схеми електричної принципової;

– розробку алгоритму роботи та програмного забезпечення для мікроконтролера та сервера;

3 Вимоги до системи

3.1 Вимоги до системи в цілому

Комп’ютерна система табелювання робочого часу на основі RFID-технології повинна забезпечити:

1. Автоматичне зчитування інформації про працівника підприємства на основі RFID-технології;

2. Сповіщення користувача про успішність / невдачу зчитування інформації або передачі даних;

3. Відображення переданої інформації на сайті;

4. Безвідмовну роботу при температурі повітря навколишнього середовища від -5 °С до +85 °С, при відносній вологості повітря до 90%.

3.1.1 Вимоги до структури та функціонування системи

Структура системи табелювання робочого часу на основі RFID-технології включає в себе:

– однокристальний мікроконтролер, який забезпечує загальне керування функціонуванням системи;

– модуль RFID, рідкокристалічний дисплей;

– віддалений сервер.

В загальному випадку, структура системи повинна реалізовувати функції табелювання робочого часу на підприємстві.

Основні функціональні вимоги характеризуються наступними критеріями:

– точність зчитування;

– надійність;

– захищеність;

– зручність монтажу та модернізації;

– коректність відображення інформації.

3.1.2 Вимоги до способів та засобів зв’язку між компонентами системи

Обмін даними між компонентами системи табелювання робочого часу на основі RFID-технології повинен здійснюватися з використанням безпровідних технологій передачі інформації.

3.1.3 Вимоги до режимів функціонування системи

Система повинна функціонувати у безперервному режимі роботи. У безперервному режимі роботи система очікує на зчитування RFID-мітки та при успішному зчитуванні надсилає дані на віддалений сервер.

3.1.4 Перспективи розвитку та модернізації системи

Передбачаються перспективи розвитку пристрою, що включають масштабування та інтеграцію в систему технології зчитування відпечатків пальців працівників.

3.1.5 Вимоги до надійності системи

Система повинна бути захищена від фізичних чи механічних пошкоджень на рівні апаратного та програмного забезпечення. Надійність системи повинна забезпечувати відновлюваність функціонування у випадку збою апаратного чи програмного забезпечення.

Показники надійності системи табелювання робочого часу на основі RFID-технології повинні відповідати вимогам ДСТУ 3396.0-96. Ймовірність безвідмовної роботи системи повинна складати не менше 99,6%

3.1.6 Вимоги до функцій та задач, які виконує система

Функції та задачі, які повинна виконувати система, передбачають:

– зчитування інформації про працівника на підприємстві з допомогою технології RFID;

– автоматична передача зчитаних даних про працівника на віддалений сервер;

– дистанційний моніторинг робочого часу працівників на підприємстві.

3.1.7 Вимоги до апаратного забезпечення

Вимоги до елементної бази розробки:

– режими роботи і умови експлуатації вибраних елементів повинні відповідати вказаним в ТЗ;

– вибрана елементна база має забезпечувати та реалізовувати необхідні функціональні вимоги і задачі;

– елементна база по можливості має бути широковживаною, доступною і дешевою. Необхідно також враховувати можливість заміни вибраних елементів на аналогічні (вітчизняні чи імпортного виробництва).

Вимоги до мікроконтролера:

– мікроконтролер має підтримувати RISC архітектуру команд;

– мікроконтролер повинен містити необхідний набір вбудованих периферійних пристроїв (таймери, АЦП і т.п.) та потрібну кількість керованих портів введення / виведення.

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ.

Комплект конструкторської документації повинен складатись з:

– пояснювальної записки;

– графічного матеріалу:

1. функціональна схема системи;
2. структурна схема системи;
3. схема електрична принципова;
4. блок-схема алгоритму роботи програми;

*Примітка: в комплект конструкторської документації можуть вноситися зміни та доповнення в процесі розробки.

5 Техніко-економічні показники

Собівартість розробки системи повинна становити не більше 5000 грн.

Термін експлуатації системи повинен бути не менший 10 років.

*Примітка: собівартість системи може змінюватись під час розрахунку в процесі розробки.

6 Стадії та етапи проєктування

Таблиця 1 – Стадії та етапи виконання КРБ

№ етапу	Назва етапу виконання КРБ	Термін виконання
1	Розробка та затвердження технічного завдання	18.01-21.01
2	Аналіз технічного завдання та обґрунтування можливих рішень	22.01-20.02
3	Розробка структурної та функціональної схем	21.02-19.03
4	Розробка програмного забезпечення для проєктованої системи	20.03-20.04
5	Розробка схеми електричної принципової, вибір елементної бази	21.04-15.05
6	Опрацювання питань розділу «Безпека життєдіяльності, основи охорони праці»	16.05-28.05
7	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	29.05-05.06
8	Оформлення графічної частини	06.06-11.06
9	Попередній захист кваліфікаційної роботи	12.06-18.06
10	Захист кваліфікаційної роботи бакалавра	19.06-23.06

7 Додаткові умови виконання кваліфікаційної роботи бакалавра

Під час виконання кваліфікаційної роботи бакалавра в дане технічне завдання можуть вноситися зміни та доповнення.

Додаток Б

Структури таблиць бази даних

Розроблена таблиця «employee» та її поля наведено в таблиці Б.1.

Таблиця Б.1 – emoloyee

Індекс	Назва поля	Тип даних
AI PK	employeeId	int
	cardCode	varchar(255)
	employeeName	varchar(255)
	employeePosition	varchar(255)
	employeeStatus	varchar(255)
	endTime	time
	startTime	time

Розроблена таблиця «entry» та її поля наведено в таблиці Б.2.

Таблиця Б.2 – entry

Індекс	Назва поля	Тип даних
AI PK	entryId	int
FK	employee	int
	entryTime	datetime(6)
	entryType	varchar(255)

Розроблена таблиця «user» та її поля наведено в таблиці Б.3.

Таблиця Б.3 – user

Індекс	Назва поля	Тип даних
AI PK	userId	int
UK	userEmail	varchar(255)
FK	employee	int
	username	varchar(255)
	userPassword	varchar(255)
	userRole	varchar(255)

Розроблена таблиця «workDay» та її поля наведено в таблиці Б.4.

Таблиця Б.4 – workDay

Індекс	Назва поля	Тип даних
AI PK	dayId	int
FK	employee	int
	date	date
	dayStatus	varchar(255)

Додаток В

Перелік елементів

Поз. позначення	*Найменування*		*Примітка*
	Мікросхеми		
U1	Мікроконтролер ESP32 DevKit V1	1	
U2	RFID-модуль RC522	1	
U3	I2C-конвертер на PCF 8574T	1	
U4	LCD QC1602A	1	
	Світлодіоди		
LED1	RGB-світлодіод SKV149	1	
	П'єзодинаміки		
LS1	Активний п'єзодинамік KY-012	1	
	Резистори		
R1, R2	0805-0,125-200 Ом±5%	2	

					КС КРБ 123.357.00.00 ПЕ	
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розроб.		*Мельник Н.О.*			*Комп'ютерна система табелювання робочого часу на основі RFID-технології*	*Літ.* / *Арк.* 75 / *Аркушів* 1
Перевір.		*Тиш Є. В.*				
Реценз.						*ТНТУ, каф. КС, гр. СІс-41*
Н. Контр.		*Луцик Н. С.*			*Перелік елементів*	
Зав. каф.		*Осухівська Г.М.*				

Додаток Г

Лістинг програми мікроконтролера

Лістинг Г.1 – Код програми мікроконтролера ESP32 Devkit V1.

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <SPI.h>
#include <MFRC522.h>
#include "time.h"
#include "LiquidCrystal_I2C.h"

#define RedRGB_Pin 12
#define GreenRGB_Pin 14
#define Buzzer_Pin 13
#define SS_Pin 5
#define RST_Pin 15

const char* Ssid = "MikroTik-DD086E";
const char* Password = "";
const char* NtpServer = "pool.ntp.org";
String serverName =
"http://192.168.88.23:8080/webController?command=espData&uid=
";

LiquidCrystal_I2C lcd(0x27, 16, 2);
MFRC522 mfrc522(SS_Pin, RST_Pin);

hw_timer_t* timer = NULL;
volatile SemaphoreHandle_t timerSemaphore;

void IRAM_ATTR onTimer() {
  xSemaphoreGiveFromISR(timerSemaphore, NULL);
}

void setup() {
  pinMode(RedRGB_Pin, OUTPUT);
  pinMode(GreenRGB_Pin, OUTPUT);
  pinMode(Buzzer_Pin, OUTPUT);
  digitalWrite(RedRGB_Pin, LOW);
  digitalWrite(GreenRGB_Pin, LOW);
  digitalWrite(Buzzer_Pin, LOW);

  Serial.begin(115200);

  lcd.init();
  lcd.backlight();

  SPI.begin();
```

```
  mfrc522.PCD_Init();
  delay(4);

  wifiConnect();

  initTime("EET-2EEST,M3.5.0/3,M10.5.0/4");

  timerSemaphore = xSemaphoreCreateBinary();
  timer = timerBegin(0, 80, true);
  timerAttachInterrupt(timer, &onTimer, true);
  timerAlarmWrite(timer, 1000000, true);
  timerAlarmEnable(timer);
}

void loop() {
  if (xSemaphoreTake(timerSemaphore, 0) == pdTRUE) {
    printLocalTime();
  }

  digitalWrite(RedRGB_Pin, HIGH);
  digitalWrite(GreenRGB_Pin, LOW);

  if (!mfrc522.PICC_IsNewCardPresent() ||
!mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  // Dump UID
  String strUID = "";
  Serial.print(F("Card UID: "));
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    strUID.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  mfrc522.PICC_HaltA();

  digitalWrite(RedRGB_Pin, LOW);
  digitalWrite(GreenRGB_Pin, HIGH);
  digitalWrite(Buzzer_Pin, HIGH);
  delay(200);
  digitalWrite(Buzzer_Pin, LOW);
  delay(200);
  digitalWrite(Buzzer_Pin, HIGH);
  delay(100);
  digitalWrite(Buzzer_Pin, LOW);

  sendGET(strUID);
}

void wifiConnect() {
  digitalWrite(RedRGB_Pin, HIGH);
```

```
  digitalWrite(GreenRGB_Pin, HIGH);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("No connection...");
  WiFi.begin(Ssid, Password);
  Serial.println(F("Connecting"));
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print(F("Connected to WiFi network with IP Address:
"));
  Serial.println(WiFi.localIP());
}

void initTime(String timezone) {
  struct tm timeinfo;

  Serial.println("Setting up time...");
  configTime(0, 0, NtpServer);
  Serial.printf("Setting Timezone to %s\n", timezone.c_str());
  setenv("TZ", timezone.c_str(), 1);
  tzset();

  if (!getLocalTime(&timeinfo)) {
    printErrorGetTime();
    return;
  }
  Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S zone %Z %z");

  printLocalTime();
}

void printLocalTime(void) {
  struct tm timeinfo;

  if (!getLocalTime(&timeinfo)) {
    printErrorGetTime();
    return;
  }

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(&timeinfo, "%b %d %Y");
  lcd.setCursor(0, 1);
  lcd.print(&timeinfo, "%a %H:%M:%S");
}

void printErrorGetTime(void) {
  Serial.println("Failed to obtain time");
  lcd.clear();
  lcd.setCursor(0, 0);
```

```
  lcd.print("Failed to");
  lcd.setCursor(0, 1);
  lcd.print("obtain time");
}

void sendGET(String strUID) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Sending data...");

  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;

    String serverPath = serverName + strUID;

    int httpResponseCode = http.GET();

    if (httpResponseCode > 0) {
      Serial.print(F("HTTP Response code: "));
      Serial.println(httpResponseCode);
      String payload = http.getString();
      Serial.println(payload);
    } else {
      Serial.print(F("Error code: "));
      Serial.println(httpResponseCode);

      digitalWrite(RedRGB_Pin, HIGH);
      digitalWrite(GreenRGB_Pin, LOW);
      lcd.clear();
      lcd.setCursor(3, 0);
      lcd.print("Error: ");
      lcd.print(String(httpResponseCode));
      lcd.setCursor(0, 1);
      lcd.print("Data wasn`t sent");
      delay(3000);
    }
    http.end();
  } else {
    Serial.println(F("WiFi Disconnected"));
    wifiConnect();
  }
}
```

Додаток Д

Лістинги файлів серверної частини системи

Лістинг Д.1 – Лістинг файлу «WebController.java».

```
package ua.tntu.server.controller;

import ua.tntu.server.dao.EmployeeDAO;
import ua.tntu.server.dao.EntryDAO;
import ua.tntu.server.dao.UserDAO;
import ua.tntu.server.dao.WorkDayDAO;
import ua.tntu.server.model.Employee;
import ua.tntu.server.model.Entry;
import ua.tntu.server.model.User;
import ua.tntu.server.service.EmployeeService;
import ua.tntu.server.service.EntryService;
import ua.tntu.server.service.UserService;
import ua.tntu.server.service.WorkDayService;
import ua.tntu.server.util.ScheduleCheck;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.util.Calendar;
import java.util.Date;
import java.util.Timer;

public class WebController extends HttpServlet {
    private Timer timer;
    private String commandTemplate="webController?command=";

    public void init() {
        timer = new Timer();
        Calendar calendar = Calendar.getInstance();
        calendar.set(Calendar.HOUR_OF_DAY, 18);
        calendar.set(Calendar.MINUTE, 0);
        calendar.set(Calendar.SECOND, 0);
        Date timeToRun = calendar.getTime();
        timer.schedule(new ScheduleCheck(), timeToRun,
86400000);
    }

    @Override
    protected void doGet(HttpServletRequest req,
HttpServletResponse resp)
            throws ServletException, IOException {
```

```
        req.getRequestDispatcher(process(req)).forward(req,
resp);
    }

    @Override
    protected void doPost(HttpServletRequest req,
HttpServletResponse resp)
            throws ServletException, IOException {
        resp.sendRedirect(process(req));
    }

    private String process(HttpServletRequest request) {
        EmployeeService employeeService = new
EmployeeService(new EmployeeDAO());
        EntryService entryService = new EntryService(new
EntryDAO());
        UserService userService = new UserService(new
UserDAO());
        WorkDayService workDayService = new WorkDayService(new
WorkDayDAO());

        String commandName = request.getParameter("command");
        if(commandName == null){
            return "logIn.jsp";
        }

        if(commandName.equals("espData")){
            String code = request.getParameter("uid");
            Employee employee =
employeeService.getEmployeeByCode(code);

employee.setEmployeeStatus(employee.getEmployeeStatus().equals("
Active") ? "Absent" : "Active");
            entryService.addEntry(new Entry(
                    employee,
                    LocalDateTime.now(),
                    employee.getEmployeeStatus()
            ));
        }

        if(commandName.equals("logIn")){
            String login = request.getParameter("email");
            String password = request.getParameter("password");

            User user = userService.getUserByLogin(login);
            if(user != null &&
user.getUserPassword().equals(password)){
                request.getSession().setAttribute("user", user);
                return commandTemplate+"main";
            }
        }

        if(commandName.equals("logOut")){
```

```
            request.getSession().invalidate();
            return "logIn.jsp";
        }

        if(commandName.equals("main")){
            if(request.getSession().getAttribute("user") != null
&&

((User)(request.getSession().getAttribute("user"))).getUserRole(
).equals("Admin")){

                request.setAttribute("employees",
employeeService.getAllEmployee());

                return "main.jsp";
            }else if(request.getSession().getAttribute("user")
!= null &&

((User)(request.getSession().getAttribute("user"))).getUserRole(
).equals("User")){
                return commandTemplate + "employeeInfo" + "&id="
                        +
((User)request.getSession().getAttribute("user")).getEmployee().
getEmployeeId();
            }else {
                return "logIn.jsp";
            }
        }

        if(commandName.equals("employeeInfo")){
            int employeeId =
Integer.parseInt(request.getParameter("id"));

            request.setAttribute("employee", employeeService
                    .getEmployee(employeeId));
            request.setAttribute("entries", entryService
                    .getAllEntryByEmployee(employeeId));
            request.setAttribute("workDays", workDayService
                    .getAllEntryByEmployee(employeeId));

            return "employeeInfo.jsp";
        }

        if(commandName.equals("accountInfo")){
            return "accountInfo.jsp";
        }

        if(commandName.equals("signUp")){
            if(request.getMethod().equals("GET")){
                return "signUp.jsp";
            }else {
                String userEmail =
request.getParameter("email");
```

```
                String userPassword =
request.getParameter("password");
                String userName = request.getParameter("name");
                String position =
request.getParameter("position");
                String code = request.getParameter("code");
                String role = request.getParameter("role");
                LocalTime startTime =
LocalTime.parse(request.getParameter("startTime"));
                LocalTime endTime =
LocalTime.parse(request.getParameter("endTime"));

                Employee employee = new Employee(userName,
position, "Active", startTime, endTime, code);
                User user = new User(userName, userEmail,
userPassword, role);
                employee.setUser(user);
                user.setEmployee(employee);

                userService.addUser(user);

                return "logIn.jsp";
            }
        }

        return "logIn.jsp";
    }
}
```

Лістинг Д.2 – Лістинг файлу «EmployeeDAO.java».

```
package ua.tntu.server.dao;

import org.hibernate.Session;
import org.hibernate.Transaction;
import ua.tntu.server.model.Employee;
import ua.tntu.server.util.HibernateUtil;

import java.util.List;

public class EmployeeDAO {
    public void addEmployee(Employee employee){
        Transaction transaction = null;
        try(Session session =
HibernateUtil.getSessionFactory().openSession()){
            transaction = session.beginTransaction();
            session.save(employee);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
```

```
                transaction.rollback();
            }
            e.printStackTrace();
        }
    }

    public void updateEmployee(Employee employee){
        Transaction transaction = null;
        try(Session session =
HibernateUtil.getSessionFactory().openSession()){
            transaction = session.beginTransaction();
            session.update(employee);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
    }

    public void deleteEmployee(int id){
        Transaction transaction = null;
        try(Session session =
HibernateUtil.getSessionFactory().openSession()){
            transaction = session.beginTransaction();

            Employee employee = session.get(Employee.class, id);
            if (employee != null) {
                session.delete(employee);
            }
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
    }

    public Employee getEmployee(int id) {
        Transaction transaction = null;
        Employee employee = null;
        try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();
            employee = session.get(Employee.class, id);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
```

```
        }
        return employee;
    }

    public Employee getEmployeeByCode(String code) {
        Transaction transaction = null;
        Employee employee = null;
        try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();
            employee = (Employee) session.createQuery("from
Employee  where cardCode = :code")
                    .setParameter("code", code)
                    .getResultList().get(0);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
        return employee;
    }

    @SuppressWarnings("unchecked")
    public List<Employee> getAllEmployee() {
        Transaction transaction = null;
        List <Employee> listOfEmployee = null;
        try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();

            listOfEmployee = session.createQuery("from
Employee").getResultList();

            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
        return listOfEmployee;
    }
}
```

Лістинг Д.3 – Лістинг файлу «EntryDAO.java».

```
package ua.tntu.server.dao;
```

```
import org.hibernate.Session;
import org.hibernate.Transaction;
import ua.tntu.server.model.Entry;
import ua.tntu.server.util.HibernateUtil;

import java.sql.Date;
import java.time.LocalDate;
import java.util.List;

public class EntryDAO {
    public void addEntry(Entry entry){
        Transaction transaction = null;
        try(Session session =
HibernateUtil.getSessionFactory().openSession()){
            transaction = session.beginTransaction();
            session.save(entry);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
    }

    public void updateEntry(Entry entry){
        Transaction transaction = null;
        try(Session session =
HibernateUtil.getSessionFactory().openSession()){
            transaction = session.beginTransaction();
            session.update(entry);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
    }

    public void deleteEntry(int id){
        Transaction transaction = null;
        try(Session session =
HibernateUtil.getSessionFactory().openSession()){
            transaction = session.beginTransaction();

            Entry entry = session.get(Entry.class, id);
            if (entry != null) {
                session.delete(entry);
            }
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
```

```
                transaction.rollback();
            }
            e.printStackTrace();
        }
    }

    public Entry getEntry(int id) {
        Transaction transaction = null;
        Entry entry = null;
        try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();
            entry = session.get(Entry.class, id);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
        return entry;
    }

    @SuppressWarnings("unchecked")
    public List<Entry> getAllEntry() {
        Transaction transaction = null;
        List <Entry> listOfEntry = null;
        try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();

            listOfEntry = session.createQuery("from Entry
").getResultList();

            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
        return listOfEntry;
    }

    @SuppressWarnings("unchecked")
    public List<Entry> getAllEntryByEmployee(int employeeId) {
        Transaction transaction = null;
        List <Entry> listOfEntry = null;
        try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();

            listOfEntry = session.createQuery("from Entry where
```

```
employee = " + employeeId).getResultList();

            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
        return listOfEntry;
    }

    public List<Entry> getAllEntryByDay(int id, LocalDate date)
{
        Transaction transaction = null;
        List <Entry> listOfEntry = null;
        try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();

            listOfEntry = session.createQuery("from Entry where
employee = :id AND DATE(entryTime) = :date")
                    .setParameter("id", id)
                    .setParameter("date", Date.valueOf(date))
                    .getResultList();

            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
        return listOfEntry;
    }
}
```

Лістинг Д.4 – Лістинг файлу «UserDAO.java».

```
package ua.tntu.server.dao;

import org.hibernate.Session;
import org.hibernate.Transaction;
import ua.tntu.server.model.User;
import ua.tntu.server.util.HibernateUtil;

import java.util.List;

public class UserDAO {
    public void addUser(User user){
```

```
        Transaction transaction = null;
        try(Session session =
HibernateUtil.getSessionFactory().openSession()){
            transaction = session.beginTransaction();
            session.save(user);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
    }

    public void updateUser(User user){
        Transaction transaction = null;
        try(Session session =
HibernateUtil.getSessionFactory().openSession()){
            transaction = session.beginTransaction();
            session.update(user);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
    }

    public void deleteUser(int id){
        Transaction transaction = null;
        try(Session session =
HibernateUtil.getSessionFactory().openSession()){
            transaction = session.beginTransaction();

            User user = session.get(User.class, id);
            if (user != null) {
                session.delete(user);
            }
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
    }

    public User getUser(int id) {
        Transaction transaction = null;
        User user = null;
        try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
```

```
            transaction = session.beginTransaction();
            user = session.get(User.class, id);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
        return user;
    }

    public User getUserByLogin(String login) {
        Transaction transaction = null;
        User user = null;
        try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();
            user = session.byNaturalId(User.class)
                    .using("userEmail", login)
                    .load();
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
        return user;
    }

    @SuppressWarnings("unchecked")
    public List<User> getAllUser() {
        Transaction transaction = null;
        List <User> listOfUser = null;
        try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();

            listOfUser = session.createQuery("from
User").getResultList();

            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
        return listOfUser;
    }
}
```

Лістинг Д.5 – Лістинг файлу «WorkDayDAO.java».

```
package ua.tntu.server.dao;

import org.hibernate.Session;
import org.hibernate.Transaction;
import ua.tntu.server.model.WorkDay;
import ua.tntu.server.util.HibernateUtil;

import java.util.List;

public class WorkDayDAO {
    public void addWorkDay(WorkDay workDay){
        Transaction transaction = null;
        try(Session session =
HibernateUtil.getSessionFactory().openSession()){
            transaction = session.beginTransaction();
            session.save(workDay);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
    }

    public void updateWorkDay(WorkDay workDay){
        Transaction transaction = null;
        try(Session session =
HibernateUtil.getSessionFactory().openSession()){
            transaction = session.beginTransaction();
            session.update(workDay);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
    }

    public void deleteWorkDay(int id){
        Transaction transaction = null;
        try(Session session =
HibernateUtil.getSessionFactory().openSession()){
            transaction = session.beginTransaction();

            WorkDay workDay = session.get(WorkDay.class, id);
            if (workDay != null) {
                session.delete(workDay);
```

```
            }
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
    }

    public WorkDay getWorkDay(int id) {
        Transaction transaction = null;
        WorkDay workDay = null;
        try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();
            workDay = session.get(WorkDay.class, id);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
        return workDay;
    }

    @SuppressWarnings("unchecked")
    public List<WorkDay> getAllWorkDay() {
        Transaction transaction = null;
        List <WorkDay> listOfWorkDay = null;
        try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();

            listOfWorkDay = session.createQuery("from WorkDay
").getResultList();

            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
        return listOfWorkDay;
    }

    @SuppressWarnings("unchecked")
    public List<WorkDay> getAllWorkDayByEmployee(int employeeId)
{
        Transaction transaction = null;
        List <WorkDay> listOfWorkDay = null;
```

```
        try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();

            listOfWorkDay = session.createQuery("from WorkDay
where employee = "+employeeId).getResultList();

            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
        return listOfWorkDay;
    }
}
```

Лістинг Д.6 – Лістинг файлу «Employee.java».

```
package ua.tntu.server.model;

import javax.persistence.*;
import java.time.LocalTime;
import java.util.List;

@Entity
@Table(name = "employeeEntity")
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "employeeId")
    private int employeeId;
    @Column(name = "employeeName")
    private String employeeName;
    @Column(name = "employeePosition")
    private String employeePosition;
    @Column(name = "employeeStatus")
    private String employeeStatus;
    @Column(name = "startTime")
    private LocalTime startTime;
    @Column(name = "endTime")
    private LocalTime endTime;
    @Column(name = "cardCode")
    private String cardCode;

    @OneToOne(mappedBy = "employee")
    private User user;
    @OneToMany(mappedBy = "employee")
    private List<Entry> entryList;
```

```
    @OneToMany(mappedBy = "employee")
    private List<WorkDay> dayList;

    public Employee() {}

    public Employee(String employeeName, String
employeePosition, String employeeStatus,
                    LocalTime startTime, LocalTime endTime,
String cardCode) {
        this.employeeName = employeeName;
        this.employeePosition = employeePosition;
        this.employeeStatus = employeeStatus;
        this.startTime = startTime;
        this.endTime = endTime;
        this.cardCode = cardCode;
    }

    public int getEmployeeId() {
        return employeeId;
    }

    public void setEmployeeId(int employeeId) {
        this.employeeId = employeeId;
    }

    public String getEmployeeName() {
        return employeeName;
    }

    public void setEmployeeName(String employeeName) {
        this.employeeName = employeeName;
    }

    public String getEmployeePosition() {
        return employeePosition;
    }

    public void setEmployeePosition(String employeePosition) {
        this.employeePosition = employeePosition;
    }

    public LocalTime getStartTime() {
        return startTime;
    }

    public void setStartTime(LocalTime startTime) {
        this.startTime = startTime;
    }

    public LocalTime getEndTime() {
        return endTime;
    }
```

```
    public void setEndTime(LocalTime endTime) {
        this.endTime = endTime;
    }

    public String getEmployeeStatus() {
        return employeeStatus;
    }

    public void setEmployeeStatus(String employeeStatus) {
        this.employeeStatus = employeeStatus;
    }

    public String getCardCode() {
        return cardCode;
    }

    public void setCardCode(String cardCode) {
        this.cardCode = cardCode;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }
}
```

Лістинг Д.7 – Лістинг файлу «Entry.java».

```
package ua.tntu.server.model;

import javax.persistence.*;
import java.time.LocalDateTime;

@Entity
@Table(name = "entryEntity")
public class Entry {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "entryId")
    private int entryId;
    @Column(name = "entryTime")
    private LocalDateTime entryTime;
    @Column(name = "entryType")
    private String entryType;
    @ManyToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "employee")
    private Employee employee;
```

```
    public Entry() {}

    public Entry(Employee employee, LocalDateTime entryTime,
String entryType) {
        this.employee = employee;
        this.entryTime = entryTime;
        this.entryType = entryType;
    }

    public int getEntryId() {
        return entryId;
    }

    public void setEntryId(int entryId) {
        this.entryId = entryId;
    }

    public Employee getEmployee() {
        return employee;
    }

    public void setEmployee(Employee employee) {
        this.employee = employee;
    }

    public LocalDateTime getEntryTime() {
        return entryTime;
    }

    public void setEntryTime(LocalDateTime entryTime) {
        this.entryTime = entryTime;
    }

    public String getEntryType() {
        return entryType;
    }

    public void setEntryType(String entryType) {
        this.entryType = entryType;
    }
}
```

Лістинг Д.8 – Лістинг файлу «User.java».

```
package ua.tntu.server.model;

import org.hibernate.annotations.NaturalId;

import javax.persistence.*;
```

```
@Entity
@Table(name = "userEntity")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "userId")
    private int userId;
    @Column(name = "userName")
    private String userName;
    @NaturalId
    @Column(name = "userEmail", nullable = false, unique = true)
    private String userEmail;
    @Column(name = "userPassword")
    private String userPassword;
    @Column(name = "userRole")
    private String userRole;
    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "employee")
    private Employee employee;

    public User() {};

    public User(String userName, String userEmail,
                String userPassword, String userRole) {
        super();
        this.userName = userName;
        this.userEmail = userEmail;
        this.userPassword = userPassword;
        this.userRole = userRole;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getUserEmail() {
        return userEmail;
    }

    public void setUserEmail(String userEmail) {
```

```
        this.userEmail = userEmail;
    }

    public String getUserPassword() {
        return userPassword;
    }

    public void setUserPassword(String userPassword) {
        this.userPassword = userPassword;
    }

    public String getUserRole() {
        return userRole;
    }

    public void setUserRole(String userRole) {
        this.userRole = userRole;
    }

    public Employee getEmployee() {
        return employee;
    }

    public void setEmployee(Employee employee) {
        this.employee = employee;
    }
}
```

Лістинг Д.9 – Лістинг файлу «WorkDay.java».

```
package ua.tntu.server.model;

import javax.persistence.*;
import java.time.LocalDate;

@Entity
@Table(name = "workDayEntity")
public class WorkDay {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "dayId")
    private int dayId;
    @Column(name = "date")
    private LocalDate date;
    @Column(name = "dayStatus")
    private String dayStatus;
    @ManyToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "employee")
    private Employee employee;
```

```
    public WorkDay() {}

    public WorkDay(Employee employee, LocalDate date, String
dayStatus) {
        this.employee = employee;
        this.date = date;
        this.dayStatus = dayStatus;
    }

    public int getDayId() {
        return dayId;
    }

    public void setDayId(int dayId) {
        this.dayId = dayId;
    }

    public Employee getEmployee() {
        return employee;
    }

    public void setEmployee(Employee employee) {
        this.employee = employee;
    }

    public LocalDate getDate() {
        return date;
    }

    public void setDate(LocalDate date) {
        this.date = date;
    }

    public String getDayStatus() {
        return dayStatus;
    }

    public void setDayStatus(String dayStatus) {
        this.dayStatus = dayStatus;
    }
}
```

Лістинг Д.10 – Лістинг файлу «EmployeeService.java».

```
package ua.tntu.server.service;

import ua.tntu.server.dao.EmployeeDAO;
import ua.tntu.server.model.Employee;

import java.util.List;
```

```
public class EmployeeService {
    EmployeeDAO employeeDAO;

    public EmployeeService(EmployeeDAO employeeDAO) {
        this.employeeDAO = employeeDAO;
    }

    public void addEmployee(Employee employee){
        employeeDAO.addEmployee(employee);
    }

    public void updateEmployee(Employee employee){
        employeeDAO.updateEmployee(employee);
    }

    public void deleteEmployee(int id){
        employeeDAO.deleteEmployee(id);
    }

    public Employee getEmployee(int id) {
        return employeeDAO.getEmployee(id);
    }
    public Employee getEmployeeByCode(String code) {
        return employeeDAO.getEmployeeByCode(code);
    }
    public List<Employee> getAllEmployee() {
        return employeeDAO.getAllEmployee();
    }
}
```

Лістинг Д.11 – Лістинг файлу «EntryService.java».

```
package ua.tntu.server.service;

import ua.tntu.server.dao.EntryDAO;
import ua.tntu.server.model.Entry;

import java.time.LocalDate;
import java.util.List;

public class EntryService {
    EntryDAO entryDAO;

    public EntryService(EntryDAO entryDAO) {
        this.entryDAO = entryDAO;
    }

    public void addEntry(Entry employee){
        entryDAO.addEntry(employee);
```

```
    }

    public void updateEntry(Entry employee){
        entryDAO.updateEntry(employee);
    }

    public void deleteEntry(int id){
        entryDAO.deleteEntry(id);
    }

    public Entry getEntry(int id) {
        return entryDAO.getEntry(id);
    }

    public List<Entry> getAllEntry() {
        return entryDAO.getAllEntry();
    }

    public List<Entry> getAllEntryByEmployee(int id) {
        return entryDAO.getAllEntryByEmployee(id);
    }

    public List<Entry> getAllEntryByDate(int id, LocalDate
date){
        return entryDAO.getAllEntryByDay(id, date);
    }
}
```

Лістинг Д.12 – Лістинг файлу «UserService.java».

```
package ua.tntu.server.service;

import ua.tntu.server.dao.UserDAO;
import ua.tntu.server.model.User;

import java.util.List;

public class UserService {
    UserDAO userDAO;

    public UserService(UserDAO userDAO) {
        this.userDAO = userDAO;
    }


    public void addUser(User user){
        userDAO.addUser(user);
    }

    public void updateUser(User user){
```

```
        userDAO.updateUser(user);
    }

    public void deleteUser(int id){
        userDAO.deleteUser(id);
    }

    public User getUser(int id) {
        return userDAO.getUser(id);
    }
    public User getUserByLogin(String login) {
        return userDAO.getUserByLogin(login);
    }


    public List<User> getAllUser() {
        return userDAO.getAllUser();
    }
}
```

Лістинг Д.13 – Лістинг файлу «WorkDayService.java».

```
package ua.tntu.server.service;

import ua.tntu.server.dao.WorkDayDAO;
import ua.tntu.server.model.WorkDay;

import java.util.List;

public class WorkDayService {
    WorkDayDAO workDayDAO;

    public WorkDayService(WorkDayDAO workDayDAO) {
        this.workDayDAO = workDayDAO;
    }

    public void addEntry(WorkDay employee){
        workDayDAO.addWorkDay(employee);
    }

    public void updateEntry(WorkDay employee){
        workDayDAO.updateWorkDay(employee);
    }

    public void deleteEntry(int id){
        workDayDAO.deleteWorkDay(id);
    }

    public WorkDay getEntry(int id) {
        return workDayDAO.getWorkDay(id);
```

```
    }

    public List<WorkDay> getAllEntry() {
        return workDayDAO.getAllWorkDay();
    }

    public List<WorkDay> getAllEntryByEmployee(int id) {
        return workDayDAO.getAllWorkDayByEmployee(id);
    }
}
```

Лістинг Д.14 – Лістинг файлу «HibernateUtil.java».

```
package ua.tntu.server.util;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
import ua.tntu.server.model.Employee;
import ua.tntu.server.model.User;

public class HibernateUtil {
    private static SessionFactory sessionFactory;

    public static SessionFactory getSessionFactory(){
        if(sessionFactory == null){
            Configuration configuration = new Configuration();
            configuration.configure("hibernate.cfg.xml");

            configuration.addAnnotatedClass(User.class);
            configuration.addAnnotatedClass(Employee.class);

            sessionFactory =
configuration.buildSessionFactory();
        }
        return sessionFactory;
    }
}
```

Лістинг Д.15 – Лістинг файлу «ScheduleCheck.java».

```
package ua.tntu.server.util;

import ua.tntu.server.dao.EmployeeDAO;
import ua.tntu.server.dao.EntryDAO;
import ua.tntu.server.dao.WorkDayDAO;
import ua.tntu.server.model.Employee;
import ua.tntu.server.model.Entry;
import ua.tntu.server.model.WorkDay;
```

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.List;
import java.util.TimerTask;

public class ScheduleCheck extends TimerTask {
    public void run() {
        WorkDayDAO workDayDAO = new WorkDayDAO();
        EntryDAO entryDAO = new EntryDAO();
        EmployeeDAO employeeDAO = new EmployeeDAO();

        for(Employee employee : employeeDAO.getAllEmployee()){
            if(employee.getEmployeeStatus().equals("Active")){
                List<Entry> entryList = entryDAO

.getAllEntryByDay(employee.getEmployeeId(), LocalDate.now());
                if(entryList.size() == 0){
                    workDayDAO.addWorkDay(new WorkDay(employee,
LocalDate.now(), "Absent"));
                } else if (entryList.get(0)
                        .getEntryTime()
                        .toLocalTime()
                        .isAfter(employee.getStartTime())){
                    workDayDAO.addWorkDay(new WorkDay(employee,
LocalDate.now(), "Late"));
                } else {
                    workDayDAO.addWorkDay(new WorkDay(employee,
LocalDate.now(), "In time"));
                }
            }
        }
    }
}
```

Лістинг Д.16 – Лістинг файлу «hibernate.cfg.xml».

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-
3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</
property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost/diplomapr
ojectdb</property>
```

```
        <property
name="hibernate.connection.username">root</property>
        <property
name="hibernate.connection.password">admin292655</property>
        <property
name="hibernate.current_session_context_class">thread</property>
        <property name="hibernate.show_sql">true</property>
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</pr
operty>

        <mapping class="ua.tntu.server.model.Entry"/>
        <mapping class="ua.tntu.server.model.Employee"/>
        <mapping class="ua.tntu.server.model.User"/>
        <mapping class="ua.tntu.server.model.WorkDay"/>
    </session-factory>
</hibernate-configuration>
```

Лістинг Д.17 – Лістинг файлу «contentStyles.css».

```
.main {
    display: flex;
    align-items: center;
    justify-content: center;
    background-color: #f6fbff;
}

.main-wrapper {
    background-color: #1f76ff;
    color: white;
    padding: 30px;
    border-radius: 20px
}

.main table {
    text-align: left;
    border-collapse: separate;
    border-spacing: 3px;
    background-color: #0300df;
    color: #c0ffff;
    border: 10px solid #0300df;
    border-radius: 15px;
}

.main td a {
    display: block;
    position: relative;
    text-decoration: none;
    color: #c0ffff;
    transition: all 0.3s ease;
```

```
}

.main table a:hover {
    background-color: #0035ff;
}

.main table a:active {
    background-color: #0030e9;
}

.click-td {
    display: block;
    position: relative;
    background-color: #006dff;
    transition: all 0.3s ease;
}

.click-td:hover {
    background-color: #006dff;
}

.click-td:active {
    background-color: #0030e9;
}

.main table div {
    padding: 10px;
}

th {
    padding: 10px;
}

td {
    font-size: 0.9em;
    background-color: #1927ff;
}
```

Лістинг Д.18 – Лістинг файлу «logInStyles.css».

```
@import url('https://fonts.googleapis.com/css?family=Poppins');

html {
  background-color: #0045cb;
}

body {
  font-family: "Poppins", sans-serif;
  height: 100vh;
  margin: 0;
```

```
}

a {
  color: #0039ff;
  display: inline-block;
  text-decoration: none;
  font-weight: 400;
}

h2 {
  text-align: center;
  font-size: 14px;
  font-weight: 600;
  text-transform: uppercase;
  display: inline-block;
  margin: 20px 8px 10px 8px;
  color: #cccccc;
}


.wrapper {
  display: flex;
  align-items: center;
  flex-direction: column;
  justify-content: center;
  width: 100%;
  min-height: 100%;
  padding: 20px;
}

#formContent {
  -webkit-border-radius: 10px 10px 10px 10px;
  border-radius: 10px 10px 10px 10px;
  background: #fff;
  padding: 30px;
  width: 90%;
  max-width: 450px;
  position: relative;
  padding: 0px;
  -webkit-box-shadow: 0 30px 60px 0 rgba(0, 0, 0, 0.3);
  box-shadow: 0 30px 60px 0 rgba(0, 0, 0, 0.3);
  text-align: center;
}


h2.active {
  color: #0d0d0d;
  border-bottom: 2px solid #0039ff;
}

input[type=button],
input[type=submit],
input[type=reset] {
```

```
  background-color: #003aff;
  border: none;
  color: white;
  padding: 13px 80px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  text-transform: uppercase;
  font-size: 12px;
  -webkit-box-shadow: 0 10px 30px 0 rgba(95, 186, 233, 0.4);
  box-shadow: 0 10px 30px 0 rgba(95, 186, 233, 0.4);
  -webkit-border-radius: 5px 5px 5px 5px;
  border-radius: 5px 5px 5px 5px;
  margin: 5px 20px 40px 20px;
  -webkit-transition: all 0.3s ease-in-out;
  -moz-transition: all 0.3s ease-in-out;
  -ms-transition: all 0.3s ease-in-out;
  -o-transition: all 0.3s ease-in-out;
  transition: all 0.3s ease-in-out;
}

input[type=button]:hover,
input[type=submit]:hover,
input[type=reset]:hover {
  background-color: #0005dc;
}

input[type=button]:active,
input[type=submit]:active,
input[type=reset]:active {
  -moz-transform: scale(0.95);
  -webkit-transform: scale(0.95);
  -o-transform: scale(0.95);
  -ms-transform: scale(0.95);
  transform: scale(0.95);
}

input[type=text],
input[type=email],
input[type=password],
input[type=time],
select {
  background-color: #f6f6f6;
  border: none;
  color: #0d0d0d;
  padding: 13px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 14px;
  margin: 5px;
  width: 85%;
  border: 2px solid #f6f6f6;
```

```
  -webkit-transition: all 0.5s ease-in-out;
  -moz-transition: all 0.5s ease-in-out;
  -ms-transition: all 0.5s ease-in-out;
  -o-transition: all 0.5s ease-in-out;
  transition: all 0.5s ease-in-out;
  -webkit-border-radius: 5px 5px 5px 5px;
  border-radius: 5px 5px 5px 5px;
}

input[type=text]:focus,
input[type=email]:focus,
input[type=password]:focus,
select:focus {
  background-color: #fff;
  border-bottom: 2px solid #0039ff;
}

input[type=text]:placeholder,
input[type=email]:placeholder,
input[type=password]:placeholder {
  color: #cccccc;
}

.fadeInDown {
  -webkit-animation-name: fadeInDown;
  animation-name: fadeInDown;
  -webkit-animation-duration: 1s;
  animation-duration: 1s;
  -webkit-animation-fill-mode: both;
  animation-fill-mode: both;
}

@-webkit-keyframes fadeInDown {
  0% {
    opacity: 0;
    -webkit-transform: translate3d(0, -100%, 0);
    transform: translate3d(0, -100%, 0);
  }

  100% {
    opacity: 1;
    -webkit-transform: none;
    transform: none;
  }
}

@keyframes fadeInDown {
  0% {
    opacity: 0;
    -webkit-transform: translate3d(0, -100%, 0);
    transform: translate3d(0, -100%, 0);
  }
```

```
  100% {
    opacity: 1;
    -webkit-transform: none;
    transform: none;
  }
}

@-webkit-keyframes fadeIn {
  from {
    opacity: 0;
  }

  to {
    opacity: 1;
  }
}

@-moz-keyframes fadeIn {
  from {
    opacity: 0;
  }

  to {
    opacity: 1;
  }
}

@keyframes fadeIn {
  from {
    opacity: 0;
  }

  to {
    opacity: 1;
  }
}

.fadeIn {
  opacity: 0;
  -webkit-animation: fadeIn ease-in 1;
  -moz-animation: fadeIn ease-in 1;
  animation: fadeIn ease-in 1;

  -webkit-animation-fill-mode: forwards;
  -moz-animation-fill-mode: forwards;
  animation-fill-mode: forwards;

  -webkit-animation-duration: 1s;
  -moz-animation-duration: 1s;
  animation-duration: 1s;
}

.fadeIn.first {
```

```
  -webkit-animation-delay: 0.4s;
  -moz-animation-delay: 0.4s;
  animation-delay: 0.4s;
}

.fadeIn.second {
  -webkit-animation-delay: 0.6s;
  -moz-animation-delay: 0.6s;
  animation-delay: 0.6s;
}

.fadeIn.third {
  -webkit-animation-delay: 0.8s;
  -moz-animation-delay: 0.8s;
  animation-delay: 0.8s;
}

.fadeIn.fourth {
  -webkit-animation-delay: 1s;
  -moz-animation-delay: 1s;
  animation-delay: 1s;
}

.fadeIn.fifth {
  -webkit-animation-delay: 1.2s;
  -moz-animation-delay: 1.2s;
  animation-delay: 1.2s;
}

.fadeIn.sixth {
  -webkit-animation-delay: 1.4s;
  -moz-animation-delay: 1.4s;
  animation-delay: 1.4s;
}

.fadeIn.seventh {
  -webkit-animation-delay: 1.6s;
  -moz-animation-delay: 1.6s;
  animation-delay: 1.6s;
}

.fadeIn.eighth {
  -webkit-animation-delay: 1.8s;
  -moz-animation-delay: 1.8s;
  animation-delay: 1.8s;
}

.fadeIn.ninth {
  -webkit-animation-delay: 2s;
  -moz-animation-delay: 2s;
  animation-delay: 2s;
}
```

```
.fadeIn.tenth {
  -webkit-animation-delay: 2.2s;
  -moz-animation-delay: 2.2s;
  animation-delay: 2.2s;
}

.underlineHover:after {
  display: block;
  left: 0;
  bottom: -10px;
  width: 0;
  height: 2px;
  background-color: #0039ff;
  content: "";
  transition: width 0.2s;
}

.underlineHover:hover:after {
  width: 100%;
}

*:focus {
  outline: none;
}

#icon {
  width: 60%;
}

* {
  box-sizing: border-box;
}
```

Лістинг Д.19 – Лістинг файлу «navStyles.css».

```
@import url('https://fonts.googleapis.com/css?family=Poppins');

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    display: flex;
    flex-direction: column;
    font-family: "Poppins", sans-serif;
    font-size: 18px;
}
```

```
.wrapper {
    display: flex;
}

.vert-nav {
    display: flex;
    flex-direction: column;
    width: 200px;
    height: 100vh;
    background: #1f76ff;
    position: sticky;
    top: 0;
}

.vert-nav h2 {
    margin: 4vh 20px;
    text-align: center;
    color: #c0ffff;
}

.vert-nav .menu {
    display: flex;
    flex-direction: column;
    padding: 0 5px;
}

.vert-nav .menu a {
    position: relative;
    text-decoration: none;
    color: #c0ffff;
    transition: all 0.3s ease;
}

.vert-nav .menu a div::after {
    content: "";
    background: #c0ffff;
    height: 100%;
    width: 2px;
    position: absolute;
    left: 0;
    bottom: 0px;
    transform: scaleY(0);
    transition: all 0.3s ease;
}

.vert-nav .menu a div:hover::after {
    transform: scaleY(1);
}

.vert-nav .menu a .item {
    font-size: 0.9em;
    padding: 10px 20px;
}
```

```
.vert-nav .menu a:hover {
    border-radius: 15px;
    background-color: #0035ff;
}

.vert-nav .menu a:active {
    border-radius: 15px;
    background-color: #0030e9;
    transform: scale(0.95);
}

.vert-nav .footer {
    position: absolute;
    bottom: 10px;
    margin-left: 20px;
    font-size: 0.5em;
}

.main {
    flex-grow: 1;
    padding: 20px 10px;
}

.vert-nav {
    -webkit-animation-name: vert-nav;
    animation-name: vert-nav;
    -webkit-animation-duration: 1s;
    animation-duration: 1s;
    -webkit-animation-fill-mode: both;
    animation-fill-mode: both;
}

@-webkit-keyframes vert-nav {
    0% {
        opacity: 0;
        -webkit-transform: translate3d(-100%, 0, 0);
        transform: translate3d(-100%, 0, 0);
    }

    100% {
        opacity: 1;
        -webkit-transform: none;
        transform: none;
    }
}

@keyframes vert-nav {
    0% {
        opacity: 0;
        -webkit-transform: translate3d(-100%, 0, 0);
        transform: translate3d(-100%, 0, 0);
    }
```

```
    100% {
        opacity: 1;
        -webkit-transform: none;
        transform: none;
    }
}

.main-wrapper, .main-wrapper-admin {
    -webkit-animation-name: main-wrapper;
    animation-name: main-wrapper;
    -webkit-animation-duration: 1.5s;
    animation-duration: 1.5s;
    -webkit-animation-fill-mode: both;
    animation-fill-mode: both;
}

@-webkit-keyframes main-wrapper {
    0% {
        opacity: 0;
        -webkit-transform: translate3d(0, -100%, 0);
        transform: translate3d(0, -100%, 0);
    }

    100% {
        opacity: 1;
        -webkit-transform: none;
        transform: none;
    }
}

@keyframes main-wrapper {
    0% {
        opacity: 0;
        -webkit-transform: translate3d(0, -100%, 0);
        transform: translate3d(0, -100%, 0);
    }

    100% {
        opacity: 1;
        -webkit-transform: none;
        transform: none;
    }
}
```

Лістинг Д.20 – Лістинг файлу «web.xml».

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app>
```

```
  <display-name>Archetype Created Web Application</display-name>

  <servlet>
    <servlet-name>WebController</servlet-name>
    <servlet-
class>ua.tntu.server.controller.WebController</servlet-class>
    <init-param>
      <param-name>development</param-name>
      <param-value>true</param-value>
    </init-param>
  </servlet>

  <servlet-mapping>
    <servlet-name>WebController</servlet-name>
    <url-pattern>/webController</url-pattern>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
</web-app>
```

Лістинг Д.21 – Лістинг файлу «accountInfo.jsp».

```
<%@ page contentType="text/html;charset=UTF-8" language="java"
%>
<%@ taglib prefix="c"  uri="http://java.sun.com/jsp/jstl/core"
%>
<%@ taglib prefix="fn"
uri="http://java.sun.com/jsp/jstl/functions" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"
%>

<!DOCTYPE html>
<html>
<head>
    <title>Account Info</title>
    <style><%@include file="/css/navStyles.css"%></style>
    <style><%@include file="/css/contentStyles.css"%></style>
</head>
<body>
    <div class="wrapper">
        <nav class="vert-nav">
            <h2>Navigation<br>bar</h2>
            <div class="menu">
                <c:if test="${sessionScope.user.userRole eq
'Admin'}">
                    <a
href="${pageContext.request.contextPath}/webController?command=m
ain">
                        <div class="item">Main</div>
                    </a>
                </c:if>
```

```
                <!-- <c:if test="${sessionScope.user.userRole eq
'User'}">
                    <a
href="${pageContext.request.contextPath}/webController?command=e
mployeeInfo&id=${requestScope.employee.employeeId}">
                        <div class="item">Main</div>
                    </a>
                </c:if> -->
                <a
href="${pageContext.request.contextPath}/webController?command=a
ccountInfo">
                    <div class="item">Account Info</div>
                </a>
                <a
href="${pageContext.request.contextPath}/webController?command=l
ogOut">
                    <div class="item">Log Out</div>
                </a>
            </div>
            <footer class="footer">
                <p>
                    &copy;Powered by<br>
                    Nazarii Melnyk
                </p>
            </footer>
        </nav>

        <main class="main">
            <div class="main-wrapper">
                <h2>Hello, ${sessionScope.user.userName}</h2>
                <h5>Email - ${sessionScope.user.userEmail}</h5>
                <h5>Role - ${sessionScope.user.userRole}</h5>
            </div>
        </main>
    </div>
</body>
</html>
```

Лістинг Д.22 – Лістинг файлу «employeeInfo.jsp».

```
<%@ page contentType="text/html;charset=UTF-8" language="java"
%>
<%@ taglib prefix="c"  uri="http://java.sun.com/jsp/jstl/core"
%>
<%@ taglib prefix="fn"
uri="http://java.sun.com/jsp/jstl/functions" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"
%>

<!DOCTYPE html>
```

```
<html>
<head>
    <title>Employee Info</title>
    <style><%@include file="/css/navStyles.css"%></style>
    <style><%@include file="/css/contentStyles.css"%></style>
</head>
<body>
    <div class="wrapper">
        <nav class="vert-nav">
            <h2>Navigation<br>bar</h2>
            <div class="menu">
                <a
href="${pageContext.request.contextPath}/webController?command=m
ain">
                    <div class="item">Main</div>
                </a>
                <a
href="${pageContext.request.contextPath}/webController?command=a
ccountInfo">
                    <div class="item">Account Info</div>
                </a>
                <a
href="${pageContext.request.contextPath}/webController?command=l
ogOut">
                    <div class="item">Log Out</div>
                </a>
            </div>
            <footer class="footer">
                <p>
                    &copy;Powered by<br>
                    Nazarii Melnyk
                </p>
            </footer>
        </nav>

        <main class="main">
            <div class="main-wrapper">
                <h2>Employee:
${requestScope.employee.employeeName}</h2>
                <h4>Position -
${requestScope.employee.employeePosition}</h4>
                <h4>Current status -
${requestScope.employee.employeeStatus}</h4>
                <h4>Shift start time -
${requestScope.employee.startTime}</h4>
                <h4>Shift end time -
${requestScope.employee.endTime}</h4>
                <h4>Card code -
${requestScope.employee.cardCode}</h4>

                <h2>Attendance entries: </h2>
                <table>
                    <tr>
```

```
                        <th>Time</th>
                        <th>Type</th>
                    </tr>
                    <c:forEach var="entry"
items="${requestScope.entries}">
                        <tr>
                            <td>
                                <div>
                                    ${entry.entryTime}
                                </div>
                            </td>
                            <td>
                                <div>
                                    ${entry.entryType}
                                </div>
                            </td>
                        </tr>
                    </c:forEach>
                </table>

                <h2>Per day status: </h2>
                <table>
                    <tr>
                        <th>Date</th>
                        <th>Status</th>
                    </tr>
                    <c:forEach var="day"
items="${requestScope.workDays}">
                        <tr>
                            <td>
                                <div>
                                    ${day.date}
                                </div>
                            </td>
                            <td>
                                <div>
                                    ${day.dayStatus}
                                </div>
                            </td>
                        </tr>
                    </c:forEach>
                </table>
            </div>
        </main>
    </div>
</body>
</html>
```

Лістинг Д.23 – Лістинг файлу «logIn.jsp».

```
<%@ page contentType="text/html;charset=UTF-8" language="java"
%>
<%@ taglib prefix="c"  uri="http://java.sun.com/jsp/jstl/core"
%>
<%@ taglib prefix="fn"
uri="http://java.sun.com/jsp/jstl/functions" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"
%>

<!DOCTYPE html>
<html>
<head>
    <title>Log In</title>
    <style><%@include file="/css/logInStyles.css"%></style>
</head>
<body>
    <div class="wrapper fadeInDown">
        <div id="formContent">
            <h2 class="active"> Log In </h2>

            <div class="fadeIn first">
                <img src="https://cdn-icons-
png.flaticon.com/512/5087/5087607.png" id="icon" alt="User Icon"
/>
            </div>

            <form
action="${pageContext.request.contextPath}/webController?command
=logIn" method="post">
                <input type="email"
                       id="login"
                       class="fadeIn second"
                       name="email"
                       placeholder="Email"
                       value="${requestScope.email}"
                       required
                />
                <input type="password"
                       id="password"
                       class="fadeIn third"
                       name="password"
                       placeholder="Password"
                       required
                />
                <input type="submit"
                       class="fadeIn fourth"
                       value="Log In"
                />
            </form>
```

```
        </div>
    </div>
</body>
</html>
```

Лістинг Д.24 – Лістинг файлу «main.jsp».

```
<%@ page contentType="text/html;charset=UTF-8" language="java"
%>
<%@ taglib prefix="c"  uri="http://java.sun.com/jsp/jstl/core"
%>
<%@ taglib prefix="fn"
uri="http://java.sun.com/jsp/jstl/functions" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"
%>

<!DOCTYPE html>
<html>
<head>
    <title>Main</title>
    <style><%@include file="/css/navStyles.css"%></style>
    <style><%@include file="/css/contentStyles.css"%></style>
</head>
<body>
    <div class="wrapper">
        <nav class="vert-nav">
            <h2>Navigation<br>bar</h2>
            <div class="menu">
                <a
href="${pageContext.request.contextPath}/webController?command=m
ain">
                    <div class="item">Main</div>
                </a>
                <a
href="${pageContext.request.contextPath}/webController?command=s
ignUp">
                    <div class="item">Create User</div>
                </a>
                <a
href="${pageContext.request.contextPath}/webController?command=a
ccountInfo">
                    <div class="item">Account Info</div>
                </a>
                <a
href="${pageContext.request.contextPath}/webController?command=l
ogOut">
                    <div class="item">Log Out</div>
                </a>
            </div>
            <footer class="footer">
```

```
                <p>
                    &copy;Powered by<br>
                    Nazarii Melnyk
                </p>
            </footer>
        </nav>

        <main class="main">
            <div class="main-wrapper-admin">
                <table>
                    <tr>
                        <th>Employee</th>
                        <th>Status</th>
                    </tr>
                    <c:forEach var="employee"
items="${requestScope.employees}">
                        <tr>
                            <td class="click-td">
                                <a
href="${pageContext.request.contextPath}/webController?command=e
mployeeInfo&id=${employee.employeeId}">
                                    <div>
                                        ${employee.employeeName}
                                    </div>
                                </a>
                            </td>
                            <td>
                                <div>
                                    ${employee.employeeStatus}
                                </div>
                            </td>
                        </tr>
                    </c:forEach>
                </table>
            </div>
        </main>
    </div>
</body>
</html>
```

Лістинг Д.25 – Лістинг файлу «signUp.jsp».

```
<%@ page contentType="text/html;charset=UTF-8" language="java"
%>
<%@ taglib prefix="c"  uri="http://java.sun.com/jsp/jstl/core"
%>
<%@ taglib prefix="fn"
uri="http://java.sun.com/jsp/jstl/functions" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"
%>
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Sign Up</title>
    <style><%@include file="/css/logInStyles.css"%></style>
</head>
<body>
    <div class="wrapper fadeInDown">
        <div id="formContent">
            <h2 class="active"> User creation form </h2>

            <form
action="${pageContext.request.contextPath}/webController?command
=signUp" method="post">
                <input type="text"
                       id="login"
                       class="fadeIn first"
                       name="name"
                       placeholder="Name"
                       required
                />
                <input type="email"
                       id="email"
                       class="fadeIn second"
                       name="email"
                       placeholder="Email@example.com"
                       required
                />
                <input type="password"
                       id="password"
                       class="fadeIn third"
                       name="password"
                       placeholder="Password"
                       required
                />
                <input type="password"
                       id="confirmPassword"
                       class="fadeIn fourth"
                       name="login"
                       placeholder="Confirm password"
                       required
                />
                <input type="text"
                       class="fadeIn fifth"
                       placeholder="Position"
                       name="position"
                       required
                />
                <input type="text"
                       class="fadeIn sixth"
                       placeholder="Card UID"
                       name="code"
```

```
                    required
                />
                <select name="role" class="fadeIn seventh">
                    <option value="Admin">Admin</option>
                    <option value="User">User</option>
                </select>
                <div>
                    <label class="fadeIn eighth">Start time:
</label>
                    <input type="time"
                           class="fadeIn eighth"
                           min="06:00"
                           max="21:00"
                           name="startTime"
                           required
                    />
                </div>
                <div>
                    <label class="fadeIn ninth">End time:
</label>
                    <input type="time"
                           class="fadeIn ninth"
                           min="06:00"
                           max="21:00"
                           name="endTime"
                           required
                    />
                </div>
                <input type="submit"
                       class="fadeIn tenth"
                       value="Create user"
                />
            </form>
        </div>
    </div>
</body>
</html>
```

Лістинг Д.26 – Лістинг файлу «DiplomaPR.iml».

```
<?xml version="1.0" encoding="UTF-8"?>
<module version="4">
  <component name="FacetManager">
    <facet type="hibernate" name="Hibernate">
      <configuration>
        <datasource-map>
          <unit-entry name="hibernate.cfg.xml" />
        </datasource-map>
        <naming-strategy-map />
        <deploymentDescriptor name="hibernate.cfg.xml"
```

```
url="file://$MODULE_DIR$/src/main/resources/hibernate.cfg.xml"
/>
      </configuration>
    </facet>
  </component>
</module>
```

Лістинг Д.27 – Лістинг файлу «pom.xml».

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.example</groupId>
    <artifactId>DiplomaPR</artifactId>
    <packaging>war</packaging>
    <properties>
        <maven.compiler.source>19</maven.compiler.source>
        <maven.compiler.target>19</maven.compiler.target>
    </properties>
    <version>1.0-SNAPSHOT</version>
    <name>DiplomaPR Maven Webapp</name>
    <url>http://maven.apache.org</url>
    <dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>javax.servlet.jsp-api</artifactId>
            <version>2.3.3</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>5.6.15.Final</version>
        </dependency>
        <dependency>
            <groupId>com.mysql</groupId>
```

```
            <artifactId>mysql-connector-j</artifactId>
            <version>8.0.32</version>
        </dependency>
        <dependency>
            <groupId>jstl</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>
    </dependencies>
    <build>
        <finalName>DiplomaPR</finalName>
    </build>
</project>
```