**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

**Факультет прикладної математики**

**Кафедра системного програмування і спеціалізованих комп’ютерних систем**

**ЛАБОРАТОРНА РОБОТА №2**

***з дисципліни “Web-дизайн”***

**ТЕМА: “Розробка функціональності Web-додатка мовою Javascript”**

**Група: КВ-22**

**Виконав: Савельєв О.С**

**Київ – 2025**

**Загальна інформація**

*Завдання на лабораторну роботу:*

розробити статичні Web-сторінки додатку та перевірити коректне їх відображення на екранах різного розміру.

*Посилання на Github-репозиторій:*

[AlexandrCC/WEB\_LR1 (github.com)](https://github.com/AlexandrCC/WEB_LR1)

*Посилання на Github Pages:*

<https://alexandrcc.github.io/WEB_LR1/>

*Контакт студента в Telegram:*

<https://t.me/H1mars>

*Назва додатку:*

Pomadoro timer

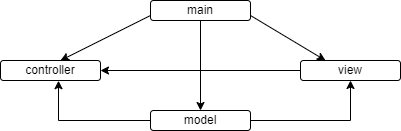
*Короткий опис його функціональності:*

Веб-додаток, який реалізує таймер помадоро для підвищення продуктивності роботи. Також на сайті можна розмістити список задач які необхіно виконати і відслідковувати кількість робочих секцій по методу помадоро

*Схема переходу:*

З будь-якої сторінки сайту можна перейти на будь-яку іншу сайту.

**Діаграма модулів Javascript додатку**



Стрілочка веде від модуля, що імпортує, до модуля, що експортується.

**Скріни усіх сторінок Web-додатка**

*Робоча сторінка*

Изображение выглядит как текст, программное обеспечение, Операционная система, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

*Профіль*

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

*Про додаток*

*Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.*

*Сторінка реєстрації*

*Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.*

*Сторінка входу*

*Изображение выглядит как текст, программное обеспечение, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.*

**Код Javascript-файлів**

*main.js*

// js/main.js

import { TimerModel, TaskModel } from "./model.js";

import { TimerView, TaskView } from "./view.js";

import { AppController } from "./controller.js";

document.addEventListener("DOMContentLoaded", () => {

// Check if we're on pomodoro.html

const timerEl = document.getElementById("timer");

const sessionEl = document.getElementById("sessionCounter");

const taskListEl = document.getElementById("taskList");

if (timerEl && sessionEl && taskListEl) {

// We are on pomodoro.html

const timerModel = new TimerModel();

const taskModel = new TaskModel();

const timerView = new TimerView(timerEl, sessionEl);

const taskView = new TaskView(taskListEl);

const controller = new AppController(timerModel, taskModel, timerView, taskView);

controller.init();

}

});

*Controller.js*

// js/controller.js

import { TimerModel, TaskModel } from "./model.js";

import { TimerView, TaskView } from "./view.js";

export class AppController {

constructor(timerModel, taskModel, timerView, taskView) {

this.timerModel = timerModel;

this.taskModel = taskModel;

this.timerView = timerView;

this.taskView = taskView;

this.timer = null;

this.decrementTime = this.decrementTime.bind(this);

this.handleAddTask = this.handleAddTask.bind(this);

this.handleEditTask = this.handleEditTask.bind(this);

this.handleDeleteTask = this.handleDeleteTask.bind(this);

this.onStart = this.onStart.bind(this);

this.onReset = this.onReset.bind(this);

}

init() {

// Render initial timer, sessions, tasks

this.timerView.renderTimer(this.timerModel.remaining);

this.timerView.renderSessions(this.timerModel.sessionCounter);

this.taskView.render(this.taskModel.tasks);

// Set up event listeners

const btnStart = document.getElementById("startTimerBtn");

const btnReset = document.getElementById("resetTimerBtn");

const btnAddTask = document.getElementById("addTaskBtn");

const taskList = document.getElementById("taskList");

const taskInput = document.getElementById("taskInput");

if (btnStart) btnStart.addEventListener("click", this.onStart);

if (btnReset) btnReset.addEventListener("click", this.onReset);

if (btnAddTask) btnAddTask.addEventListener("click", this.handleAddTask);

// Listen for custom events from TaskView

if (taskList) {

taskList.addEventListener("editTask", this.handleEditTask);

taskList.addEventListener("deleteTask", this.handleDeleteTask);

}

// Optionally handle "Enter" key for adding tasks

if (taskInput) {

taskInput.addEventListener("keypress", (e) => {

if (e.key === "Enter") {

this.handleAddTask();

}

});

}

}

decrementTime() {

if (this.timerModel.decrement()) {

this.timerView.renderTimer(this.timerModel.remaining);

} else {

clearInterval(this.timer);

this.timerView.showAlert("Time's up! Take a break.");

this.timerModel.incrementSession();

this.timerModel.reset();

this.timerView.renderTimer(this.timerModel.remaining);

this.timerView.renderSessions(this.timerModel.sessionCounter);

}

}

onStart() {

clearInterval(this.timer);

this.timer = setInterval(this.decrementTime, 1000);

}

onReset() {

clearInterval(this.timer);

this.timerModel.reset();

this.timerView.renderTimer(this.timerModel.remaining);

}

handleAddTask() {

const taskInput = document.getElementById("taskInput");

if (!taskInput) return;

const val = taskInput.value.trim();

if (val) {

this.taskModel.addTask(val);

this.taskView.render(this.taskModel.tasks);

taskInput.value = "";

}

}

handleEditTask(e) {

// e.detail.idx, e.detail.newText

const idx = e.detail.idx;

const newText = e.detail.newText;

this.taskModel.editTask(idx, newText);

this.taskView.render(this.taskModel.tasks);

}

handleDeleteTask(e) {

const idx = e.detail.idx;

this.taskModel.deleteTask(idx);

this.taskView.render(this.taskModel.tasks);

}

}

*model.js*

*// js/model.js*

*export class TimerModel {*

*constructor() {*

*this.duration = 25 \* 60; // 25 minutes*

*this.remaining = this.duration;*

*this.sessionCounter = parseInt(localStorage.getItem("pomSessions")) || 0;*

*}*

*decrement() {*

*if (this.remaining > 0) {*

*this.remaining--;*

*return true;*

*}*

*return false;*

*}*

*reset() {*

*this.remaining = this.duration;*

*}*

*incrementSession() {*

*this.sessionCounter++;*

*localStorage.setItem("pomSessions", this.sessionCounter);*

*}*

*}*

*export class TaskModel {*

*constructor() {*

*this.tasks = JSON.parse(localStorage.getItem("pomTasks")) || [];*

*}*

*addTask(txt) {*

*this.tasks.push(txt);*

*this.save();*

*}*

*editTask(idx, newText) {*

*this.tasks[idx] = newText;*

*this.save();*

*}*

*deleteTask(idx) {*

*this.tasks.splice(idx, 1);*

*this.save();*

*}*

*save() {*

*localStorage.setItem("pomTasks", JSON.stringify(this.tasks));*

*}*

*}*

*view.js*

*// js/view.js*

*export class TimerView {*

*constructor(timerEl, sessionEl) {*

*this.timerEl = timerEl;*

*this.sessionEl = sessionEl;*

*}*

*renderTimer(remaining) {*

*const minutes = Math.floor(remaining / 60);*

*const seconds = remaining % 60;*

*this.timerEl.textContent = minutes + ":" + (seconds < 10 ? "0" : "") + seconds;*

*}*

*renderSessions(count) {*

*this.sessionEl.textContent = "Completed sessions: " + count;*

*}*

*showAlert(msg) {*

*alert(msg);*

*}*

*}*

*export class TaskView {*

*constructor(listEl) {*

*this.listEl = listEl;*

*}*

*render(tasks) {*

*this.listEl.innerHTML = "";*

*tasks.forEach((task, idx) => {*

*const li = document.createElement("li");*

*li.textContent = task;*

*const btnDiv = document.createElement("div");*

*const btnEdit = document.createElement("button");*

*btnEdit.textContent = "Edit";*

*btnEdit.style.marginRight = "6px";*

*btnEdit.addEventListener("click", () => {*

*const newText = prompt("Edit your task:", task);*

*if (newText && newText.trim()) {*

*// Pass control back to the controller (which we'll handle in controller)*

*// Or we can do a simpler approach here if we have direct access to model*

*// But let's do the simpler approach: We fire an event?*

*// Instead, for minimal example, we directly modify tasks array*

*// But let's keep it minimal and pass the event upwards.*

*const evt = new CustomEvent("editTask", {*

*bubbles: true,*

*detail: { idx, newText: newText.trim() }*

*});*

*this.listEl.dispatchEvent(evt);*

*}*

*});*

*const btnDel = document.createElement("button");*

*btnDel.textContent = "Delete";*

*btnDel.addEventListener("click", () => {*

*const evt = new CustomEvent("deleteTask", {*

*bubbles: true,*

*detail: { idx }*

*});*

*this.listEl.dispatchEvent(evt);*

*});*

*btnDiv.appendChild(btnEdit);*

*btnDiv.appendChild(btnDel);*

*li.appendChild(btnDiv);*

*this.listEl.appendChild(li);*

*});*

*}*

*}*