МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Институт математики и информационных систем
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Дата сдачи на проверку:

«__»_____ 2025 г.

Проверено:

«__»_____ 2025 г.

Разработка клиент–серверного веб-приложения на HTML, CSS, JavaScript и Express.js.

Отчёт по лабораторной работе №2
по дисциплине
«Технологии Программирования»

Разработал студент гр. ИВТб-2301-05-00 _____/Черкасов А. А./
(подпись)

Преподаватель _____/Пащенко Д. Э./
(подпись)

Киров
2025

Цели лабораторной работы

- изучить структуру клиент–серверных веб-приложений с файловой загрузкой;
- освоить работу с Express.js, multer для обработки файлов и JSON хранилищем;
- реализовать систему голосования с защитой от повторного голосования;
- разработать интерактивную галерею изображений с модальными окнами;
- закрепить навыки работы с асинхронными запросами и DOM манипуляциями.

Задание

Создать веб-приложение «Meme Repo» — репозиторий мемов с системой голосования, удовлетворяющее требованиям:

1. На стороне клиента реализовать две HTML-страницы: загрузки и галереи изображений.
2. Стили разместить в отдельном файле CSS, логику обработки — в отдельных JS файлах.
3. На стороне сервера реализовать Express.js с multer для обработки загрузки файлов.
4. Реализовать JSON-хранилище для метаданных изображений и голосов.
5. Создать систему голосования с защитой от повторного голосования по IP.
6. Организовать вывод топ-10 изображений для фонового отображения.
7. Реализовать модальные окна для просмотра изображений в полном размере.

Ход выполнения работы

1. Создание структуры проекта

Проект «Meme Repo» был создан в соответствии с архитектурой веб-приложения:

- `index.html` — страница загрузки мемов с формой;
- `gallery.html` — страница галереи с системой голосования;
- `style.css` — стили оформления для обеих страниц;
- `script.js` — клиентская логика для формы загрузки;
- `server.js` — сервер Express.js с API для загрузки и голосования;
- `images.json` — хранилище метаданных изображений;
- `votes.json` — хранилище данных голосования;
- `public/uploads/` — директория для загруженных изображений.

2. Реализация клиентской части

Были разработаны две HTML-страницы с единой системой стилей:

- **Страница загрузки** (`index.html`): форма с полями имени, email и выбора файла изображения;
- **Страница галереи** (`gallery.html`): динамически загружаемая галерея с кнопками голосования;
- **Модальные окна**: полноразмерный просмотр изображений с возможностью закрытия;
- **Фоновые элементы**: анимированные миниатюры топ-10 мемов для атмосферы.

JavaScript реализует асинхронные запросы к API сервера для загрузки изображений и голосования.

3. Реализация серверной части

Сервер был создан на Express.js с использованием следующих модулей:

- `express` — основной веб-фреймворк;

- `multer` — middleware для обработки multipart/form-data (загрузка файлов);
- `fs/path` — работа с файловой системой.

На сервере реализована следующая функциональность:

- **API эндпоинты**:
  - `GET /images` — получение топ-10 изображений для фона;
  - `GET /gallery` — получение всех изображений с метаданными;
  - `POST /vote` — голосование за изображение (с защитой от повторного голосования);
  - `POST /` — загрузка нового изображения.
- **Файловое хранилище**: сохранение изображений в `public/uploads/` с уникальными именами;
- **JSON-хранилище**: метаданные изображений и голоса хранятся в JSON-файлах;
- **Защита от спама**: IP-based ограничение голосования (один голос на изображение);
- **Обработка ошибок**: валидация данных и корректные HTTP-статусы.

4. Система голосования

Реализована система голосования с следующими особенностями:

- Два типа голосов: положительный и отрицательный;
- Защита от повторного голосования по IP-адресу;
- Автоматическая сортировка изображений по количеству голосов;
- Обновление интерфейса в реальном времени после голосования.

5. Тестирование работы приложения

После запуска сервера:

```
node server.js
```

Приложение доступно по адресу `http://localhost:8080`.

Были протестированы следующие сценарии:

- Загрузка изображений с метаданными;
- Просмотр галереи с сортировкой по голосам;
- Голосование за изображения (с защитой от повторного голосования);
- Полноразмерный просмотр изображений в модальных окнах;
- Анимированный фон с миниатюрами популярных мемов.

6. Скриншоты

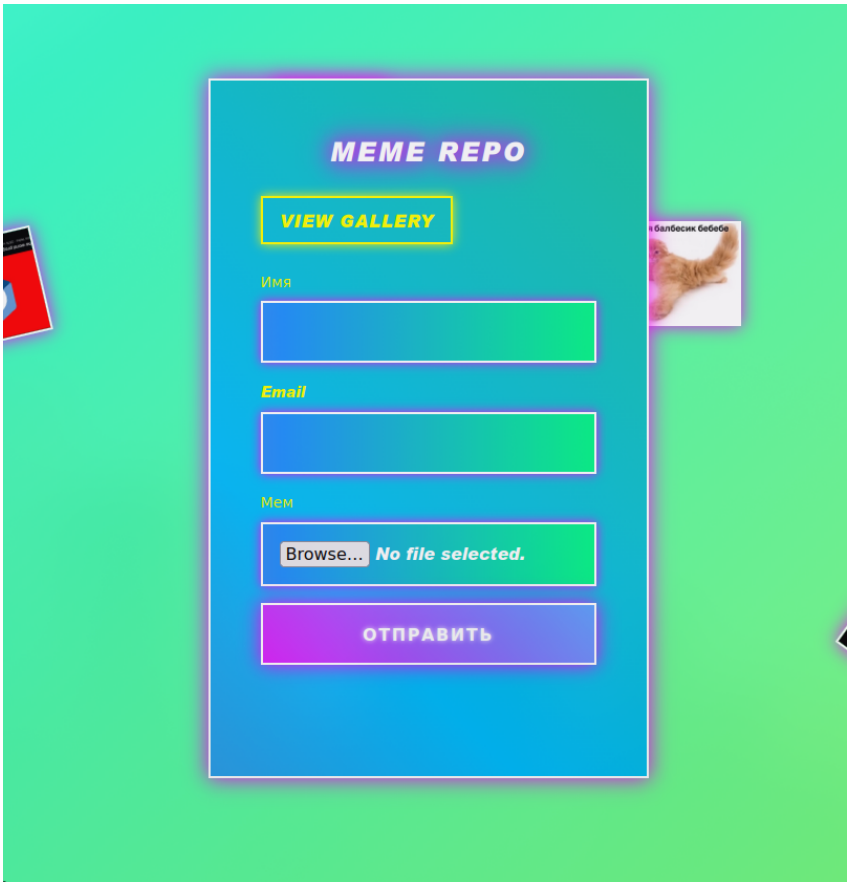

Рисунок 1 — Форма загрузки мемов

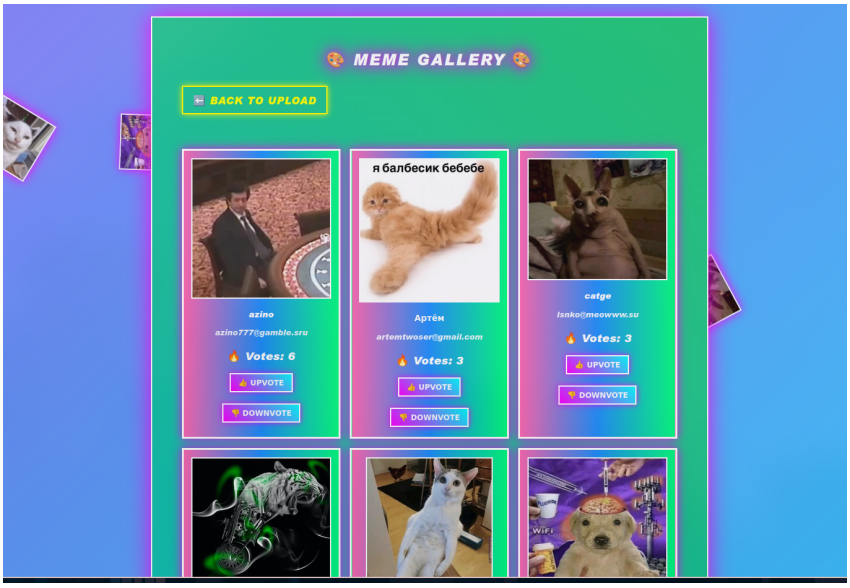

Рисунок 2 — Галлерея загруженных мемов

Вывод

В ходе выполнения лабораторной работы было разработано полнофункциональное веб-приложение «Meme Repo» с системой голосования. Были изучены современные подходы к разработке клиент-серверных приложений с файловой загрузкой.

Ключевые достижения:

- Реализована комплексная система загрузки и хранения изображений с метаданными;
- Создан интерактивный интерфейс галереи с модальными окнами и системой голосования;
- Внедрена защита от повторного голосования на основе IP-адресов;
- Организовано JSON-хранилище для персистентности данных;
- Разработан responsive дизайн с анимированными элементами интерфейса.

Работа позволила закрепить навыки работы с Express.js, асинхронными запросами, файловой системой и современными веб-технологиями. Приложение демонстрирует практическое применение архитектурных паттернов веб-разработки и принципов пользовательского интерфейса.

Приложение А1. Исходный код index.html

```
<!doctype html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Лабораторная форма</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <h1 class="title">Meme Repo</h1>
    <a href="gallery.html">View Gallery</a>

    <form id="form" enctype="multipart/form-data">
      <div class="form-group">
        <label class="form-label">Имя</label>
        <input class="form-control" name="name" required>
      </div>

      <div class="form-group">
        <label class="form-label">Email</label>
        <input class="form-control" name="email" type="email" required>
      </div>

      <div class="form-group">
        <label class="form-label">Мем</label>
        <input class="form-control" type="file" name="meme" accept="image/*" required>
      </div>

      <button class="btn">Отправить</button>
    </form>

    <div id="response" class="response"></div>
  </div>

  <div class="spinning-meme"></div>
```

```
  <div class="spinning-meme"></div>
  <div class="spinning-meme"></div>
  <div class="spinning-meme"></div>
  <div class="spinning-meme"></div>
  <div class="spinning-meme"></div>
  <div class="spinning-meme"></div>
  <div class="spinning-meme"></div>

  <script src="script.js"></script>
</body>
</html>
```

Приложение А2. Исходный код gallery.html

```
<!doctype html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Meme Gallery</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container gallery-container">
    <h1 class="title"> Meme Gallery </h1>
    <a href="/"> Back to Upload</a>
    <div id="gallery"></div>
    <div id="loading" style="text-align: center; padding: 20px; color: var(--text-secondar
      Loading memes...
    </div>
    <div id="empty" style="text-align: center; padding: 20px; color: var(--text-secondary)
      No memes yet! Be the first to upload!
    </div>
  </div>

  <!-- Modal for image view -->
  <div id="imageModal" class="modal">
```

```
  <span class="modal-close">&times;</span>
  <img class="modal-content" id="modalImage" alt="Meme fullsize">
</div>

<!-- Background spinning memes -->
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>

<script>
  const galleryDiv = document.getElementById('gallery');
  const loadingDiv = document.getElementById('loading');
  const emptyDiv = document.getElementById('empty');

  function loadGallery() {
    loadingDiv.style.display = 'block';
    galleryDiv.innerHTML = '';
    emptyDiv.style.display = 'none';

    fetch('/gallery')
      .then(res => res.json())
      .then(images => {
        loadingDiv.style.display = 'none';

        if (images.length === 0) {
          emptyDiv.style.display = 'block';
          return;
        }

        const cacheBust = Date.now();
        galleryDiv.innerHTML = images.map(img => `
          <div class="gallery-item">
```

```
            <img src="${img.path}?t=${cacheBust}" alt="Meme" data-image="${img.path}" one
            <p><strong>${img.uploader_name}</strong></p>
            <p style="font-size: 0.8rem; opacity: 0.9;">${img.uploader_email}</p>
            <p style="font-size: 1.1rem; margin: 10px 0;"><strong> Votes: <span id="votes
            <div style="display: flex; gap: 10px; justify-content: center; flex-wrap: wra
              <button data-id="${img.id}" data-vote="1"> UPVOTE</button>
              <button data-id="${img.id}" data-vote="-1"> DOWNVOTE</button>
            </div>
          </div>
        `).join('');

        // Add event listeners for images
        document.querySelectorAll('.gallery-item img').forEach(img => {
          img.addEventListener('click', function() {
            openModal(this.dataset.image);
          });
        });

        // Add event listeners for vote buttons
        document.querySelectorAll('.gallery-item button').forEach(btn => {
          btn.addEventListener('click', function() {
            vote(parseInt(this.dataset.id), parseInt(this.dataset.vote));
          });
        });
      })
      .catch(err => {
        console.error('Failed to load gallery:', err);
        loadingDiv.style.display = 'none';
        galleryDiv.innerHTML = '<p style="text-align: center; color: var(--text-secondary
      });
}

function vote(id, vote) {
  fetch('/vote', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ id, vote })
```

```
  })
  .then(res => {
    if (res.ok) {
      // Update votes display
      const votesSpan = document.getElementById(`votes-${id}`);
      if (votesSpan) {
        votesSpan.textContent = parseInt(votesSpan.textContent) + vote;
      }
      // Reload gallery to reflect new order
      setTimeout(() => loadGallery(), 500);
    } else {
      return res.text().then(text => {
        alert(text);
      });
    }
  })
  .catch(err => {
    console.error('Failed to vote:', err);
    alert('Failed to vote. Please try again.');
  });
}

// Load background images
function loadBackgroundImages() {
  fetch('/images')
    .then(res => res.json())
    .then(images => {
      const spinningMemes = document.querySelectorAll('.spinning-meme');

      if (!images || images.length === 0) {
        spinningMemes.forEach(meme => meme.style.display = 'none');
        return;
      }

      // Generate random positions once
      const positions = [];
      spinningMemes.forEach(() => {
```

```
        const randomTop = Math.random() * 80 + 5;
        const randomLeft = Math.random() * 80 + 5;
        positions.push({ top: randomTop, left: randomLeft });
      });

      spinningMemes.forEach((meme, index) => {
        meme.style.display = 'block';

        // Use pre-generated position
        meme.style.top = positions[index].top + '%';
        meme.style.left = positions[index].left + '%';

        const randomImage = images[Math.floor(Math.random() * images.length)];
        const cacheBust = Date.now();
        meme.innerHTML = `<img src="/${randomImage}?t=${cacheBust}" alt="Meme" style="w
        meme.style.animationDelay = (Math.random() * 10) + 's';
      });
    })
    .catch(err => console.error('Failed to load background images:', err));
}

loadGallery();
loadBackgroundImages();

// Modal functions
function openModal(imagePath) {
  const modal = document.getElementById('imageModal');
  const modalImg = document.getElementById('modalImage');
  modal.classList.add('show');
  const cacheBust = Date.now();
  modalImg.src = imagePath + '?t=' + cacheBust;
  document.body.style.overflow = 'hidden';
}

function closeModal() {
  const modal = document.getElementById('imageModal');
  modal.classList.remove('show');
```

```
      document.body.style.overflow = '';
    }

    // Event listeners for modal
    document.getElementById('imageModal').addEventListener('click', function(e) {
      if (e.target === this || e.target.classList.contains('modal-close')) {
        closeModal();
      }
    });

    document.getElementById('modalImage').addEventListener('click', function(e) {
      e.stopPropagation();
    });

    // Close modal on ESC key
    document.addEventListener('keydown', function(e) {
      if (e.key === 'Escape') {
        closeModal();
      }
    });
  </script>
</body>
</html>
```

Приложение А3. Исходный код script.js

```
const form = document.getElementById('form');
const responseBox = document.getElementById('response');
const submitBtn = form.querySelector('.btn');

const handleSubmit = async (e) => {
  e.preventDefault();
  const data = new FormData(form);
  submitBtn.disabled = true;
  submitBtn.textContent = 'Отправка...';

  try {
```

```
    const res = await fetch('/', { method: 'POST', body: data });
    const text = await res.text();

    if (res.ok) {
      responseBox.textContent = `Сервер ответил: ${text}`;
      responseBox.classList.add('show');
      form.reset();
      setTimeout(() => loadBackgroundImages(), 1000);
    } else {
      responseBox.textContent = `Ошибка сервера: ${text}`;
      responseBox.classList.add('show');
    }
  } catch (error) {
    responseBox.textContent = `Ошибка: ${error.message}`;
    responseBox.classList.add('show');
  } finally {
    submitBtn.disabled = false;
    submitBtn.textContent = 'Отправить';
  }

  setTimeout(() => responseBox.classList.remove('show'), 5000);
};

function loadBackgroundImages() {
  fetch('/images')
    .then(res => res.json())
    .then(images => {
      const spinningMemes = document.querySelectorAll('.spinning-meme');
      const positions = [];

      spinningMemes.forEach(() => {
        let randomTop, randomLeft;
        let attempts = 0;
        do {
          randomTop = Math.random() * 80 + 5;
          randomLeft = Math.random() * 80 + 5;
          attempts++;
```

```
        } while (isOverlapping(randomTop, randomLeft, positions) && attempts < 50);
        positions.push({ top: randomTop, left: randomLeft });
      });

      spinningMemes.forEach((meme, index) => {
        meme.style.display = images.length > 0 ? 'block' : 'none';
        if (images.length > 0) {
          meme.style.top = positions[index].top + '%';
          meme.style.left = positions[index].left + '%';
          const randomImage = images[Math.floor(Math.random() * images.length)];
          // Add cache busting parameter
          const cacheBust = Date.now();
          meme.innerHTML = `<img src="/${randomImage}?t=${cacheBust}" alt="Meme" style="wi
          meme.style.animationDelay = (Math.random() * 10) + 's';
          meme.style.animationDuration = (8 + Math.random() * 4) + 's';
        }
      });
    })
    .catch(err => {
      console.error('Failed to fetch images:', err);
      document.querySelectorAll('.spinning-meme').forEach(meme => meme.style.display = 'no
    });
}

function isOverlapping(top, left, positions) {
  const minDistance = 15;
  for (const pos of positions) {
    const distance = Math.sqrt(Math.pow(pos.top - top, 2) + Math.pow(pos.left - left, 2));
    if (distance < minDistance) return true;
  }
  return false;
}

if (form) {
  loadBackgroundImages();
  form.addEventListener('submit', handleSubmit);
}
```

Приложение А4. Исходный код style.css

```
:root {
  --bg-primary: linear-gradient(135deg, #ff00ff, #00ffff, #ffff00);
  --bg-secondary: linear-gradient(45deg, #ff1493, #00bfff, #32cd32);
  --bg-tertiary: linear-gradient(90deg, #ff69b4, #1e90ff, #00ff7f);
  --accent-primary: #ff00ff;
  --accent-secondary: #00ffff;
  --text-primary: #ffffff;
  --text-secondary: #ffff00;
  --border-color: #ffffff;
  --glow-color: #ff00ff;
  --font-main: 'Arial Black', 'Impact', sans-serif;
  --font-size-base: 1rem;
  --font-size-lg: 1.25rem;
  --font-size-xl: 1.5rem;
  --space-sm: 0.5rem;
  --space-md: 1rem;
  --space-lg: 1.5rem;
  --space-xl: 2rem;
  --space-xxl: 3rem;
  --radius-md: 0px;
  --radius-xl: 0px;
  --shadow-xl: 0 0 40px var(--glow-color);
  --transition-normal: all 0.25s ease;
}

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

body {
  font-family: var(--font-main);
  background: linear-gradient(135deg, #ff00ff, #00ffff, #ffff00);
```

```
  background-size: 400% 400%;
  animation: gradientShift 5s ease infinite;
  color: var(--text-primary);
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  padding: var(--space-xl);
  line-height: 1.6;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  position: relative;
  overflow-x: hidden;
}

@keyframes gradientShift {
  0% { background-position: 0% 50%; }
  50% { background-position: 100% 50%; }
  100% { background-position: 0% 50%; }
}

/* Container styles */
.container {
  width: 100%;
  max-width: 420px;
  background: linear-gradient(45deg, #ff1493, #00bfff, #32cd32);
  background-size: 300% 300%;
  animation: containerGlow 3s ease-in-out infinite alternate;
  padding: var(--space-xxl);
  border-radius: var(--radius-xl);
  box-shadow: var(--shadow-xl);
  border: 2px solid var(--border-color);
  transition: var(--transition-normal);
  position: relative;
  z-index: 100;
}
```

```css
/* Gallery container gets wider */
.container.gallery-container {
  max-width: 900px;
}

.container:hover {
  box-shadow: var(--shadow-lg);
  animation-duration: 1s;
}

@keyframes containerGlow {
  0% { box-shadow: 0 0 20px var(--glow-color); }
  100% { box-shadow: 0 0 40px var(--glow-color); }
}

/* Title styles */
.title {
  text-align: center;
  margin-bottom: var(--space-lg);
  font-size: var(--font-size-xl);
  font-weight: 900;
  letter-spacing: 0.1em;
  text-transform: uppercase;
  color: var(--text-primary);
  text-shadow: 0 0 10px var(--glow-color), 0 0 20px var(--glow-color), 0 0 30px var(--glow-
  animation: titleBlink 2s ease-in-out infinite alternate;
}

@keyframes titleBlink {
  0% { text-shadow: 0 0 10px var(--glow-color); }
  100% { text-shadow: 0 0 20px var(--glow-color), 0 0 30px var(--glow-color); }
}

/* Link styles */
a {
  display: inline-block;
  color: var(--text-secondary);
```

```
  text-decoration: none;
  padding: var(--space-sm) var(--space-md);
  border: 2px solid var(--text-secondary);
  border-radius: var(--radius-md);
  margin-bottom: var(--space-lg);
  transition: var(--transition-normal);
  font-weight: bold;
  text-transform: uppercase;
  letter-spacing: 0.05em;
  box-shadow: 0 0 10px var(--text-secondary);
}

a:hover {
  background: var(--text-secondary);
  color: #000;
  box-shadow: 0 0 20px var(--text-secondary);
  transform: scale(1.05);
}

/* Form styles */
form {
  display: flex;
  flex-direction: column;
  gap: var(--space-md);
}

.form-group {
  display: flex;
  flex-direction: column;
  gap: var(--space-sm);
}

.form-label {
  font-size: 0.875rem;
  color: var(--text-secondary);
  font-weight: 500;
  transition: var(--transition-fast);
```

```
}

.form-control {
  padding: var(--space-md);
  border-radius: var(--radius-md);
  border: 2px solid var(--border-color);
  background: linear-gradient(90deg, #ff69b4, #1e90ff, #00ff7f);
  background-size: 200% 200%;
  animation: inputGlow 4s ease-in-out infinite;
  color: var(--text-primary);
  font-family: inherit;
  font-size: var(--font-size-base);
  transition: var(--transition-normal);
  box-shadow: 0 0 10px var(--glow-color);
}

.form-control:focus {
  outline: none;
  border-color: var(--accent-primary);
  box-shadow: 0 0 20px var(--glow-color), 0 0 0 2px var(--accent-primary);
  animation-duration: 1s;
}

@keyframes inputGlow {
  0% { background-position: 0% 50%; }
  50% { background-position: 100% 50%; }
  100% { background-position: 0% 50%; }
}

/* Button styles */
.btn {
  width: 100%;
  padding: var(--space-md);
  border-radius: var(--radius-md);
  border: 2px solid var(--border-color);
  background: linear-gradient(45deg, var(--accent-primary), var(--accent-secondary));
  background-size: 200% 200%;
```

```
  animation: buttonPulse 2s ease-in-out infinite;
  color: white;
  font-family: inherit;
  font-size: var(--font-size-base);
  font-weight: 900;
  cursor: pointer;
  transition: var(--transition-normal);
  display: inline-flex;
  justify-content: center;
  align-items: center;
  gap: var(--space-sm);
  text-transform: uppercase;
  letter-spacing: 0.1em;
  box-shadow: 0 0 15px var(--glow-color);
  text-shadow: 0 0 5px var(--text-primary);
}

.btn:hover {
  animation-duration: 0.5s;
  box-shadow: 0 0 25px var(--glow-color);
  transform: scale(1.05);
}

.btn:active {
  transform: scale(0.95);
}

.btn:focus {
  outline: none;
  box-shadow: 0 0 30px var(--glow-color), 0 0 0 3px var(--accent-primary);
}

.btn:disabled {
  opacity: 0.6;
  cursor: not-allowed;
}
```

```
@keyframes buttonPulse {
  0% { background-position: 0% 50%; box-shadow: 0 0 15px var(--glow-color); }
  50% { background-position: 100% 50%; box-shadow: 0 0 25px var(--glow-color); }
  100% { background-position: 0% 50%; box-shadow: 0 0 15px var(--glow-color); }
}

/* Response styles */
.response {
  margin-top: var(--space-lg);
  padding: var(--space-md);
  border-radius: var(--radius-md);
  background: var(--bg-tertiary);
  color: var(--text-primary);
  border: 1px solid var(--border-color);
  opacity: 0;
  transform: translateY(-10px);
  transition: var(--transition-normal);
  pointer-events: none;
}

.response.show {
  opacity: 1;
  transform: translateY(0);
  pointer-events: auto;
}

/* Spinning Meme Images - SHARP EDGES, NO ROUNDED CORNERS */
.spinning-meme {
  position: fixed;
  width: 100px;
  height: 100px;
  background-size: cover;
  background-repeat: no-repeat;
  animation: spinAndFloat 8s linear infinite;
  z-index: 10;
  pointer-events: none;
  border: 2px solid var(--border-color);
```

```
  box-shadow: 0 0 15px var(--glow-color);
}

.spinning-meme img {
  width: 100%;
  height: 100%;
  object-fit: cover;
  border-radius: 0;
}

@keyframes spinAndFloat {
  0% { transform: translateY(0px) rotate(0deg) scale(1); }
  25% { transform: translateY(-20px) rotate(90deg) scale(1.1); }
  50% { transform: translateY(-40px) rotate(180deg) scale(1); }
  75% { transform: translateY(-20px) rotate(270deg) scale(0.9); }
  100% { transform: translateY(0px) rotate(360deg) scale(1); }
}

/* Gallery styles - SHARP EDGES */
#gallery {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 15px;
  margin-top: var(--space-xl);
  width: 100%;
}

.gallery-item {
  padding: 12px;
  border: 3px solid var(--border-color);
  background: var(--bg-tertiary);
  text-align: center;
  transition: var(--transition-normal);
  box-shadow: 0 0 15px var(--glow-color);
  display: flex;
  flex-direction: column;
  align-items: center;
```

```
}

.gallery-item:hover {
  box-shadow: 0 0 30px var(--glow-color);
  transform: scale(1.03);
}

.gallery-item img {
  max-width: 100%;
  max-height: 300px;
  object-fit: contain;
  margin-bottom: 10px;
  border: 2px solid var(--border-color);
}

.gallery-item p {
  margin: 4px 0;
  font-size: 0.85rem;
  word-wrap: break-word;
  width: 100%;
}

.gallery-item button {
  margin: 4px;
  padding: 6px 12px;
  border: 2px solid var(--border-color);
  background: linear-gradient(45deg, var(--accent-primary), var(--accent-secondary));
  color: white;
  font-weight: bold;
  cursor: pointer;
  transition: var(--transition-normal);
  box-shadow: 0 0 10px var(--glow-color);
  text-transform: uppercase;
  font-size: 0.75rem;
  min-width: 100px;
}
```

```
.gallery-item button:hover {
  box-shadow: 0 0 20px var(--glow-color);
  transform: scale(1.1);
}

.gallery-item button:active {
  transform: scale(0.95);
}

/* Responsive styles */
@media (max-width: 768px) {
  .container {
    padding: var(--space-xl);
  }

  .container.gallery-container {
    max-width: 100%;
  }

  .title {
    font-size: 1.3rem;
  }

  #gallery {
    grid-template-columns: repeat(2, 1fr);
  }
}

@media (max-width: 480px) {
  body {
    padding: var(--space-md);
  }

  .container {
    border-radius: var(--radius-lg);
    padding: var(--space-lg);
  }
```

```
  #gallery {
    grid-template-columns: 1fr;
  }
}

/* Scrollbar styles */
::-webkit-scrollbar {
  width: 10px;
  height: 10px;
}

::-webkit-scrollbar-track {
  background: var(--bg-secondary);
}

::-webkit-scrollbar-thumb {
  background: var(--accent-primary);
  border-radius: var(--radius-sm);
}

::-webkit-scrollbar-thumb:hover {
  background: var(--accent-secondary);
}

/* Modal styles */
.modal {
  display: none;
  position: fixed;
  z-index: 1000;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.9);
  opacity: 0;
  transition: opacity 0.3s ease;
```

```
}

.modal.show {
  display: flex;
  justify-content: center;
  align-items: center;
  opacity: 1;
}

.modal-content {
  max-width: 90%;
  max-height: 90%;
  object-fit: contain;
  border: 2px solid var(--border-color);
  box-shadow: 0 0 50px var(--glow-color);
}

.modal-close {
  position: absolute;
  top: 20px;
  right: 30px;
  color: var(--text-primary);
  font-size: 40px;
  font-weight: bold;
  cursor: pointer;
  z-index: 1001;
  transition: var(--transition-normal);
}

.modal-close:hover {
  color: var(--accent-primary);
  text-shadow: 0 0 10px var(--glow-color);
}

/* Accessibility styles */
:focus-visible {
  outline: 2px solid var(--accent-primary);
```

```
  outline-offset: 2px;
}
```

Приложение А5. Исходный код server.js

```
const express = require("express");
const path = require("path");
const multer = require("multer");
const fs = require("fs");

const app = express();
const port = 8080;

// Static files
app.use(express.static(path.join(__dirname, "public")));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Ensure uploads directory
const uploadsDir = path.join(__dirname, 'public', 'uploads');
if (!fs.existsSync(uploadsDir)) fs.mkdirSync(uploadsDir, { recursive: true });

// Multer config
const storage = multer.diskStorage({
  destination: uploadsDir,
  filename: (req, file, cb) => cb(null, Date.now() + '-' + Math.round(Math.random() * 1E9)
});
const upload = multer({ storage });

// JSON storage
const imagesFile = path.join(__dirname, 'images.json');
const votesFile = path.join(__dirname, 'votes.json');
let imagesData = fs.existsSync(imagesFile) ? JSON.parse(fs.readFileSync(imagesFile)) : [];
let votesData = fs.existsSync(votesFile) ? JSON.parse(fs.readFileSync(votesFile)) : {};

function saveImages() { fs.writeFileSync(imagesFile, JSON.stringify(imagesData)); }
function saveVotes() { fs.writeFileSync(votesFile, JSON.stringify(votesData)); }
```

```
// Routes
app.get("/images", (req, res) => res.json(imagesData.sort((a, b) => b.votes - a.votes).slic
app.get("/gallery", (req, res) => res.json(imagesData.sort((a, b) => b.votes - a.votes)));

app.post("/vote", (req, res) => {
  const { id, vote } = req.body;
  const img = imagesData.find(i => i.id == id);
  if (!img) return res.status(404).send("Not found");
  const ip = req.ip || 'unknown';
  if (!votesData[ip]) votesData[ip] = {};
  if (votesData[ip][id]) return res.status(429).send("Already voted");
  votesData[ip][id] = true;
  saveVotes();
  img.votes += vote;
  saveImages();
  res.send("OK");
});

app.post("/", upload.single('meme'), (req, res) => {
  if (!req.file) return res.status(400).send("No file");
  const newImage = {
    id: Date.now(),
    path: 'uploads/' + req.file.filename,
    uploader_name: req.body.name,
    uploader_email: req.body.email,
    votes: 0
  };
  imagesData.push(newImage);
  saveImages();
  res.send("OK");
});

app.get("/", (req, res) => res.sendFile(path.join(__dirname, "public", "index.html")));
app.get("/gallery.html", (req, res) => res.sendFile(path.join(__dirname, "public", "gallery

app.listen(port, () => console.log(`Server on http://localhost:${port}`));
```