

Введение

Чаты — модный и эффективный инструмент для организации мероприятий, тематических обсуждений. Многие предпочитают чаты традиционным группам, так как часто ответ может прийти моментально, а о каждом новом сообщении возможно получать уведомление.

Все мы привыкли к обычным чатам. Переписка в чате порой бывает довольно скучной. Поэтому можно прийти к выводу, что чаты нуждаются в расширении своего функционала. Интеграция в чат приложения «Игра на запоминание» не только поможет скрасить ожидание ответа пользователей, но и поможет развить вербальное и логическое мышление. Не стоит забывать, что помимо очевидной пользы можно прекрасно провести время ожидания ответа. Так как в современном мире человек тратит колоссальное количество времени на общение в чатах и не редко данное общение является бессмысленным, то общение нуждается в модификации.

Целью данной работы является разработка приложения, которое помогало бы людям не просто общаться в чате, а еще и получать от этого практическую пользу, путем развития своих когнитивных способностей, таких как память, концентрация и абстрактное мышление.

1.1 Описание чатов

Существует несколько разновидностей программной реализации чатов:

- HTTP или веб-чаты.
- Чаты, использующие технологию Adobe Flash.
- Программы-чаты для общения в локальных сетях
- Чаты, работающие по схеме клиент-сервер
- Чаты, работающие в одноранговых сетях.
- Чаты, использующие технологию Push.
- Полностью анонимные чаты.

Анонимные чаты отличаются от остальных видов чатов тем, что пользователю не нужно регистрироваться. Можно придумать любое имя, в том числе повторяющееся. В анонимных чатах невозможно отследить IP адрес.

Примеры данного вида чатов изображены на рисунке 1.1.

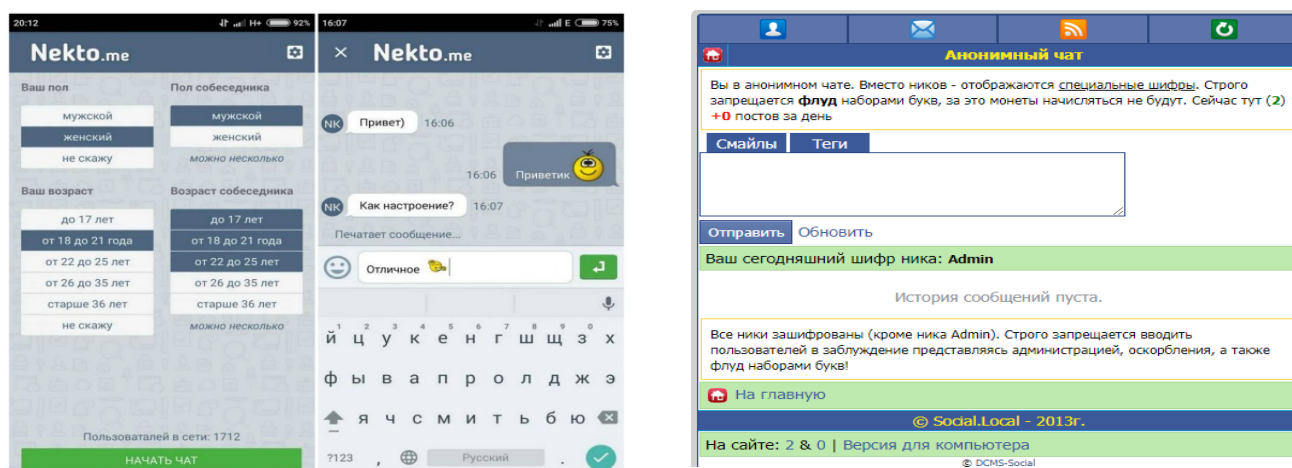


Рисунок 1.1 – Анонимные чаты

HTTP чат. Такой чат выглядит как обычная веб-страница, где можно прочесть последние несколько десятков фраз, написанные участниками чата и модераторами. Страница чата автоматически обновляется с заданной периодичностью. Изображен на рисунке 1.2.

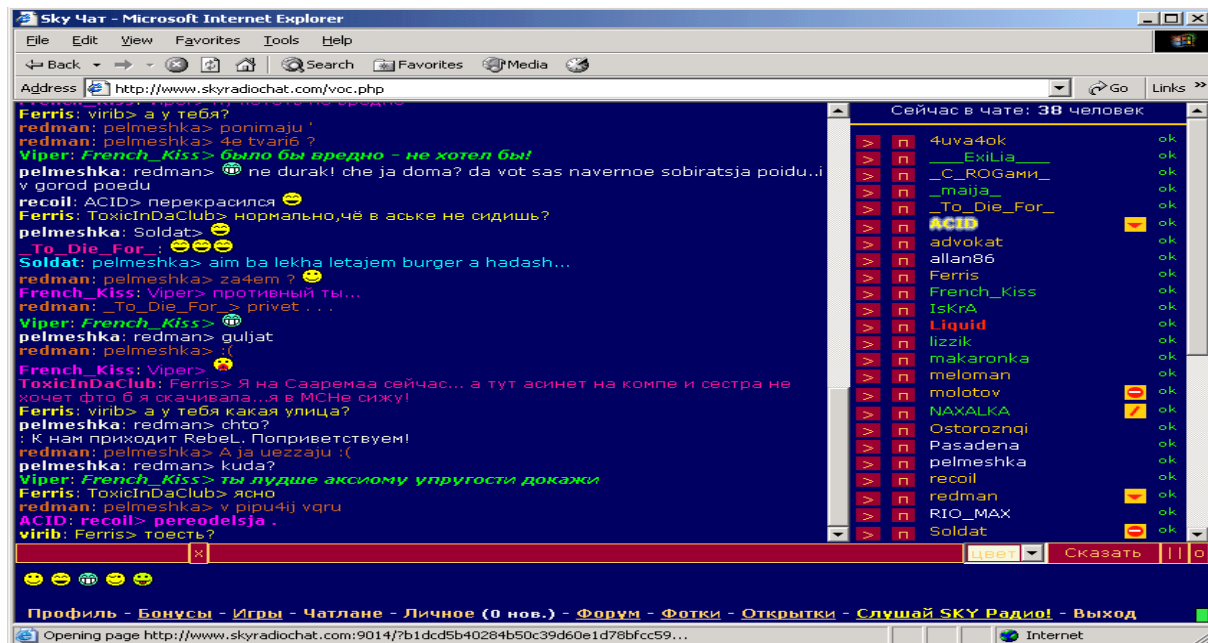


Рисунок 1.2 – HTTP чат

Чат с технологией IRC. IRC предоставляет возможность как группового, так и приватного общения.

Для группового общения существует несколько возможностей.

- Пользователь может отправить сообщение списку пользователей, при этом серверу отправляется список, сервер выделяет из него отдельных пользователей и отправляет копию сообщения каждому из них.
- Более эффективным является использование каналов. В этом случае сообщение отправляется непосредственно серверу, а сервер отправляет его всем пользователям в канале.
- Как при групповом, так и при приватном общении сообщения отправляются клиентам по кратчайшему пути и видимы только отправителю, получателю и входящим в кратчайший путь серверам.

Пример изображен на рисунке 1.3. Пример IRC-сети. Зелёным отмечены клиенты, голубым — боты, оранжевым — баунсеры.

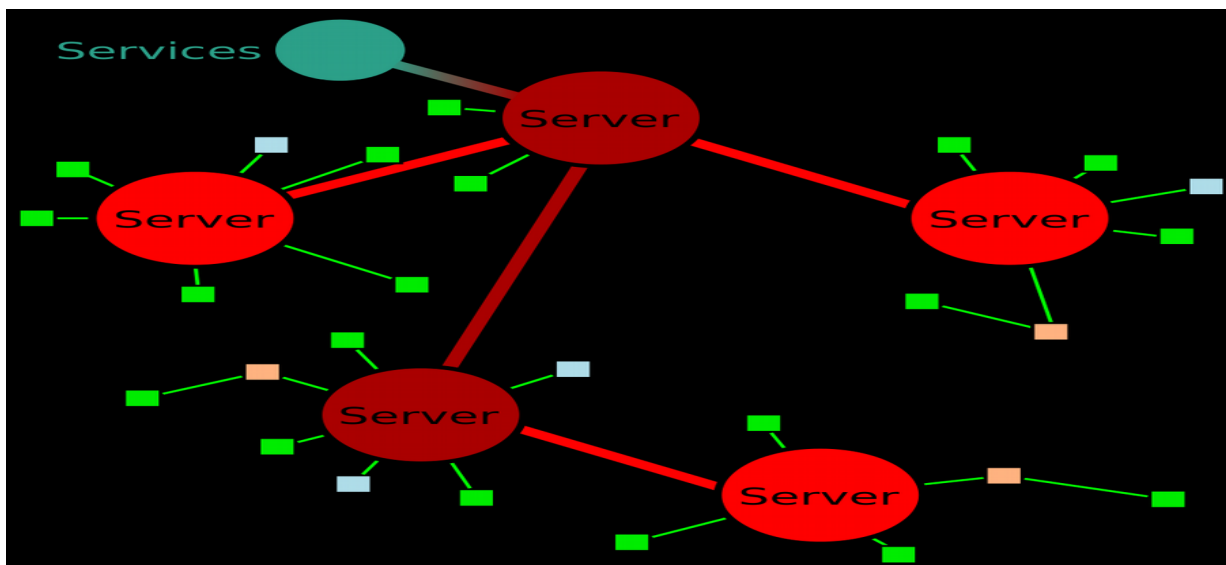


Рисунок 1.3 – Диаграмма IRC-сети

Чаты, работающие по схеме клиент-сервер. Это позволяет использовать их в сетях со сложной конфигурацией, а также управлять клиентскими приложениями (например, Mychat, Jabber).

Отсутствие дублирования кода программы-сервера программами-клиентами.

- Так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются.
- Все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще организовать контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа. Пример изображен на рисунке 1.4

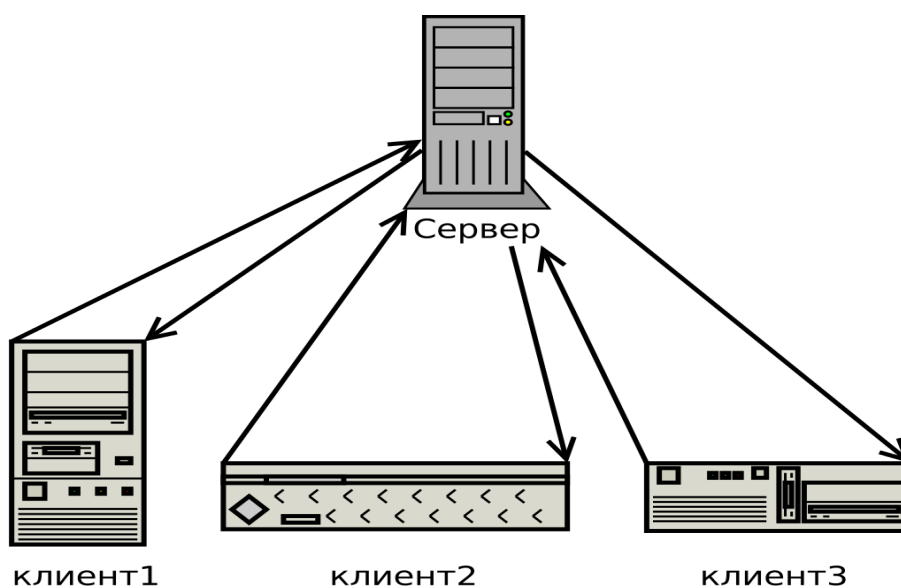


Рисунок 1.3 – Чаты, работающие по схеме клиент-сервер

1.2 Виды карточных игр

«Домино» — карточная игра. Применяется 52-карточная колода. Участвуют от 2 до 6 игроков.

Цель игры – избавиться от всех карт на руках.

Правила игры Всем участвующим в игре раздаётся по 7 карт, а оставшаяся колода кладётся закрытую рядом и верхняя карта снимается. На эту карту по очереди игрок, сидящий следом за сдающим, должен положить карту либо серию карт, которые составляют последовательности с ней, вне зависимости от масти. Например: на 7 можно положить 8 либо 6, либо 6, 5, 4, либо 8, 9, 8, 7 и т. д. Туз является «поворотной» картой: его и на него можно положить и короля, и двойку. Игрок обязан снести карту, если есть такая возможность. Если игрок не может выложить ни одну из карт, то он берёт карту из колоды. Если она подходит, то он кладёт её на кон и тянет следующую (либо выкладывает карту с рук, если теперь это возможно), если не подходит, то забирает себе и передаёт ход дальше. Выигрывает тот, кто первым избавится от своих карт на любой стадии игры, даже если остались ещё карты в колоде. Если таких нет, то считается что игра не состоялась (ничья). Эту игру относят к играм на скидывание карт.



Рисунок 1.2 – Карточное домино

«Блэкджек» — одна из самых популярных карточных игр в казино по всему миру. Большая популярность игры обуславливается простыми правилами, скоростью игры и наиболее простой стратегией в подсчёте карт. Тем не менее популярность игра завоевала не сразу. Игорным домам Соединённых Штатов приходилось стимулировать интерес к игре различными видами бонусов и выработкой нескольких разновидностей правил для блек-джека. Считается, что предшественником этой игры была карточная игра «vingt-et-un» («двадцать один»), которая появилась во французских игорных заведениях приблизительно

в XIX веке. В России, например, блек-джек по сей день часто называют двадцать одно или очко (но у традиционной игры очко несколько отличаются правила).



Рисунок 1.2 – «Блэкджек»

«Двадцать одно» — русский вариант карточной игры, известной во многих странах. Эта игра похожа на вариант игры блэкджек, в который возможно играть стандартной русской колодой (36 карт) вместо полной стандартной колоды (52 карты), так как в те времена была распространена колода из 36 карт, а колоды из 52 и 54 карт были редкостью. В отличие от блэкджека, значения карт J (валет), Q (дама) и K (король) не 10, а 2, 3 и 4 соответственно, чем частично восполняется отсутствие карт от двойки до пятёрки. Но, так как количество карт, имеющих значение 10, существенно меньше, чем в блэкджеке, то игровой баланс сильно отличается. Туз 11 либо 1. Король 4. Дама 3. Валет 2. Остальные по нумерованию.



Рисунок 1.3 – Карточная игра «Двадцать одно»

«Дурак» — карточная игра, популярная в странах бывшего СССР.

Цель игры:

В игре используется колода из 24 (сокращённая колода), 36, 52, 54 (с джокерами) карт. Участвуют от двух до шести игроков; в случае игры колодой в 52 и 54 карты может играть до 8 игроков. Старшинство карт в колоде из 24 карт (от меньшего достоинства к большему): 9, 10, В, Д, К, Т. Старшинство карт в колоде из 36 карт (от меньшего достоинства к большему): 6, 7, 8, 9, 10, В, Д, К, Т. Старшинство карт в колоде из 52 и 54 карт: 2, 3, 4, 5, 6, 7, 8, 9, 10, В, Д, К, Т. Старшинство мастей для игры в дурака не определено. Каждому раздаётся по 6 карт, следующая (или последняя, возможно и любая из колоды) карта открывается и её масть устанавливает козырь для данной игры, и оставшая колода кладётся сверху так, чтобы козырная карта была всем видна. Козырный туз в игре не отбивается. Цель игры — избавиться от всех карт. Последний игрок, не избавившийся от карт, остаётся в «дураках». Партия в дурака называется «кон». Отбитые карты идут в «отбой» («биту») и складываются на столе рубашкой вверх. Так же вы можете сыграть «вничью» если последние карты игроками были выкинуты одновременно, при таком раскладе нет ни победителей, ни проигравших.



Рисунок 1.4 – Карточная игра «Дурак»

1.3 «Игра на запоминание» описание алгоритма

«Игра на запоминание» – карточная игра, заключающаяся в нахождении одинаковых карт в «рубашке (внутренняя сторона карты. Внешняя сторона — масть карты)». В общем случае алгоритм работает следующим образом: выбирается рубашка и уровень сложности, потом запоминается уровень сложности и рубашка, создается массив в зависимости от выбранного уровня сложности, начинается игра, массив перемешивается, выкладывается массив чисел, потом на числа накладываются картинки. Каждой карте выдается id. В результате получается, что перед глазами игрока лежат карты повернутые к нему тыльной стороной, рандомно перемешанные.

Алгоритм метода «Игры на запоминание» имеет следующую последовательность действий:

- 1) выбирается рубашка
- 2) выбирается уровень сложности
- 3) запоминается уровень сложности и рубашка
- 4) создается массив в зависимости от выбранного уровня сложности
- 5) начинается игра
- 6) массив перемешивается
- 7) выкладывается массив чисел
- 8) на числа накладываются картинки

Объектно-ориентированное программирование является на сегодняшний день самой популярной, удобной и эффективной парадигмой. Поэтому процесс проектирования описанный по техническому заданию целесообразно осуществлять через призму объектно-ориентированного анализа и проектирования.

Реализуемое приложение будет состоять из HTML страницы с CSS стилями, содержащей два приложения, а также из node.js сервера, который должен обрабатывать запросы клиента, а также посылать ответ. Сервер также будет отрисовывать изображения и запускать циклы игры и соединения. Связь с сервером и манипуляции должны происходить в режиме реального времени, в связи с чем их промежуточные результаты должны оперативно отображаться на экране монитора. Дополнительные графические библиотеки не были использованы.

Пользователю будут доступны следующие возможности:

- общение в чате без регистрации.
- Разрыв соединения с сервером путем закрытия страницы;
- Выбор уровня сложности игры;
- Выбор цвета рубашки;
- Манипуляция картами;

На рисунке 3.1 продемонстрирована диаграмма прецедентов, составленная на основе анализа предоставляемых пользователю возможностей.

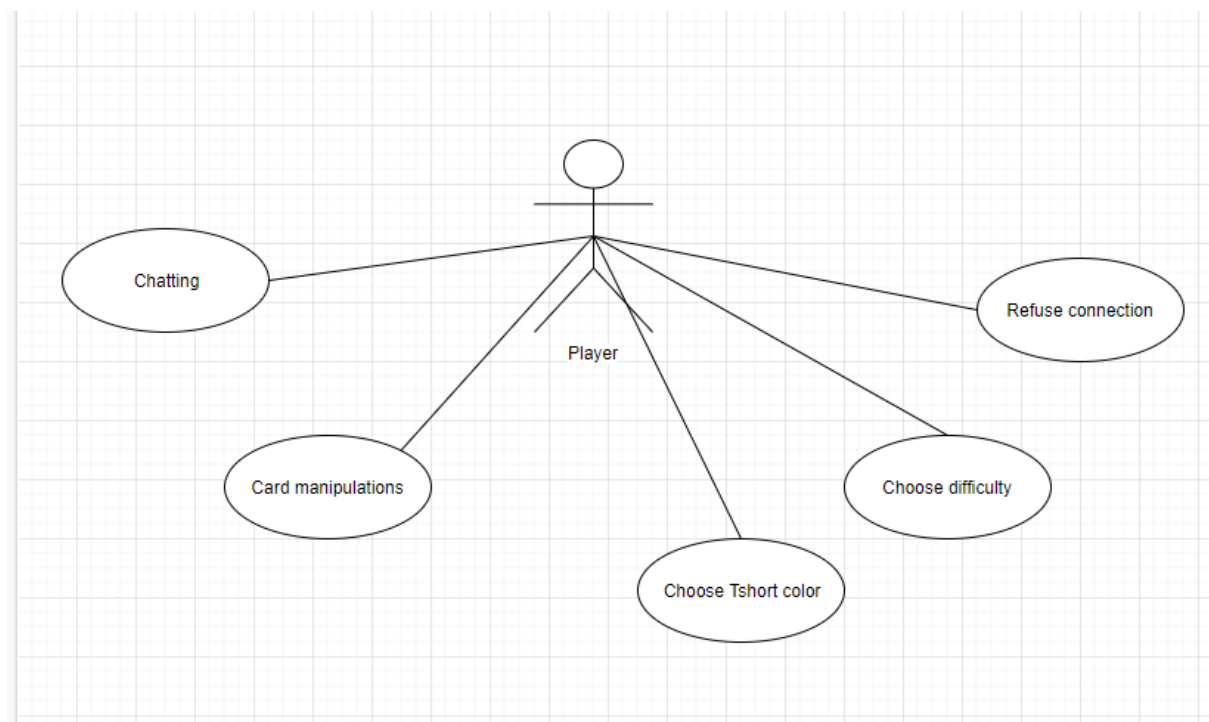


Рисунок 3.1 – Диаграмма прецедентов

Проанализировав возможности, которые должны быть доступны пользователю, можно определить, из каких компонентов должна состоять пользовательская панель инструментов:

- поле ввода имени;
- поле ввода сообщения;
- кнопка отправить;
- кнопки цвета рубашки;
- кнопки сложности;
- кнопка играть.
- кнопка выхода из игры
- кнопка переворота карты

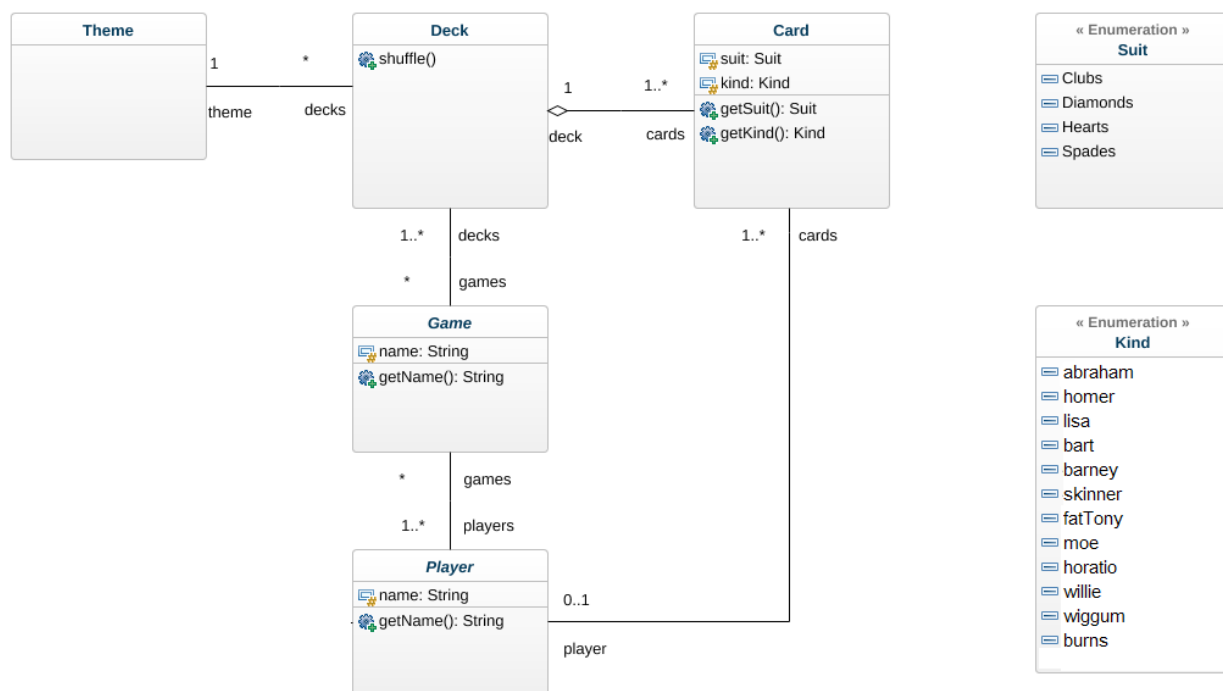


Рисунок 3.2 – Диаграмма классов

Завершение этапа проектирования системы ведёт процесс разработки к следующему этапу: этапу реализации в виде написания программного кода.

Реализация поставленной задачи предполагает предварительный выбор конкретных инструментов разработки, таких как язык программирования и библиотека для написания приложений с графическим пользовательским интерфейсом. Для этого были выбраны языки программирования «JavaScript», «HTML», «CSS» и редактор «Visual Code». А также фреймворки express, websocket, node.js .

Итог выполнения работы продемонстрирован на рисунках 4.1, 4.2, 4.3 и 4.4.

В результате реализации были удовлетворены все требования, предъявляемые к приложению на этапе создания эскизного проекта. Листинг программы приведён в приложении Б.

Последним этапом разработки является тестирование и отладка рабочего проекта. Данный этап, несмотря на то что он завершает процесс разработки, начинается на одном из промежуточных этапов реализации и проводится параллельно вместе с последним.

Для тестирования программ существуют специальные инструменты модульного тестирования, или юнит-тестирования, автоматизирующие оценку расхождения ожидаемых и реальных результатов поведения программы. Для каждого языка программирования реализован свой инструмент модульного тестирования, и, так как разработка осуществлялась с помощью языка «JavaScript», для этой цели был выбран фреймворк «Jest», предназначенный для тестирования приложений на этом языке.

Результат тестирования показал, что все обязательные функции работают без ошибок и клиент серверное приложение отображается корректно в браузерах: Safari, Chrome, Microsoft Edge, Mozilla Firefox. С помощью анализа покрытия программного кода удалось выяснить, что процент покрытия составляет 97,2%, что удовлетворяет общепринятым требованиям, согласно которым процент покрытия должен составлять более 95% от всего кода.

Заключение

В результате выполнения работы было создано клиент-серверное приложение, демонстрирующее возможность интеграции карточной игры в чат, а также позволяющее людям не просто общаться в чате, а еще и получать от этого практическую пользу, путем развития своих когнитивных способностей, таких как память, концентрация и абстрактное мышление.

При разработке курсового проекта были проделаны следующие шаги:

- анализ предметной области;
- разработка проекта;
- создание рабочего проекта;
- тестирование и отладка.

Готовый проект полностью удовлетворяет техническому заданию, а также модели, составленной на этапе создания эскизного проекта, и успешно прошёл тестирование.

Список использованных источников

1. Картинки [Электронный ресурс]. – Режим доступа:<https://www.pinterest.ru/pin/558727897503429198/?lp=true> (дата обращения:20.12.19).
2. Карточные игры [Электронный ресурс].
3. – Режим доступа:https://en.wikipedia.org/wiki/Card_game (дата обращения:05.12.19).
4. Чат (программа) [Электронный ресурс]. – Режим доступа:https://en.wikipedia.org/wiki/Online_chat :10.12.19).
5. Фреймворк express [Электронный ресурс].
6. – Режим доступа:<https://github.com/expressjs/express> :20.12.19).
7. Фреймворк socket.io [Электронный ресурс].
8. – Режим доступа:<https://github.com/socketio/socket.io> :20.12.19).
9. Фреймворк Node.js [Электронный ресурс]. –
10. Режим доступа:<https://github.com/nodejs> :20.12.19).
11. Adobe flash[Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/Adobe_Flash (Дата обращения 1.11.19)
- 12.IRC[Электронный ресурс].— Режим доступа: https://ru.wikipedia.org/wiki/Adobe_Flash (Дата обращения 12.11.19)
13. Клиент — сервер [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/Client-server_model (Дата обращения 01.12.19)
- 14.WebSocket Электронный ресурс]. — Режим доступа: <https://ru.wikipedia.org/wiki/WebSocket> (Дата обращения 12.11.19)
15. Socket.io [Электронный ресурс]. — Режим доступа: <https://socket.io/docs/> (Дата обращения 12.12.19)

Приложение А

(обязательное)

Демонстрация работы приложения

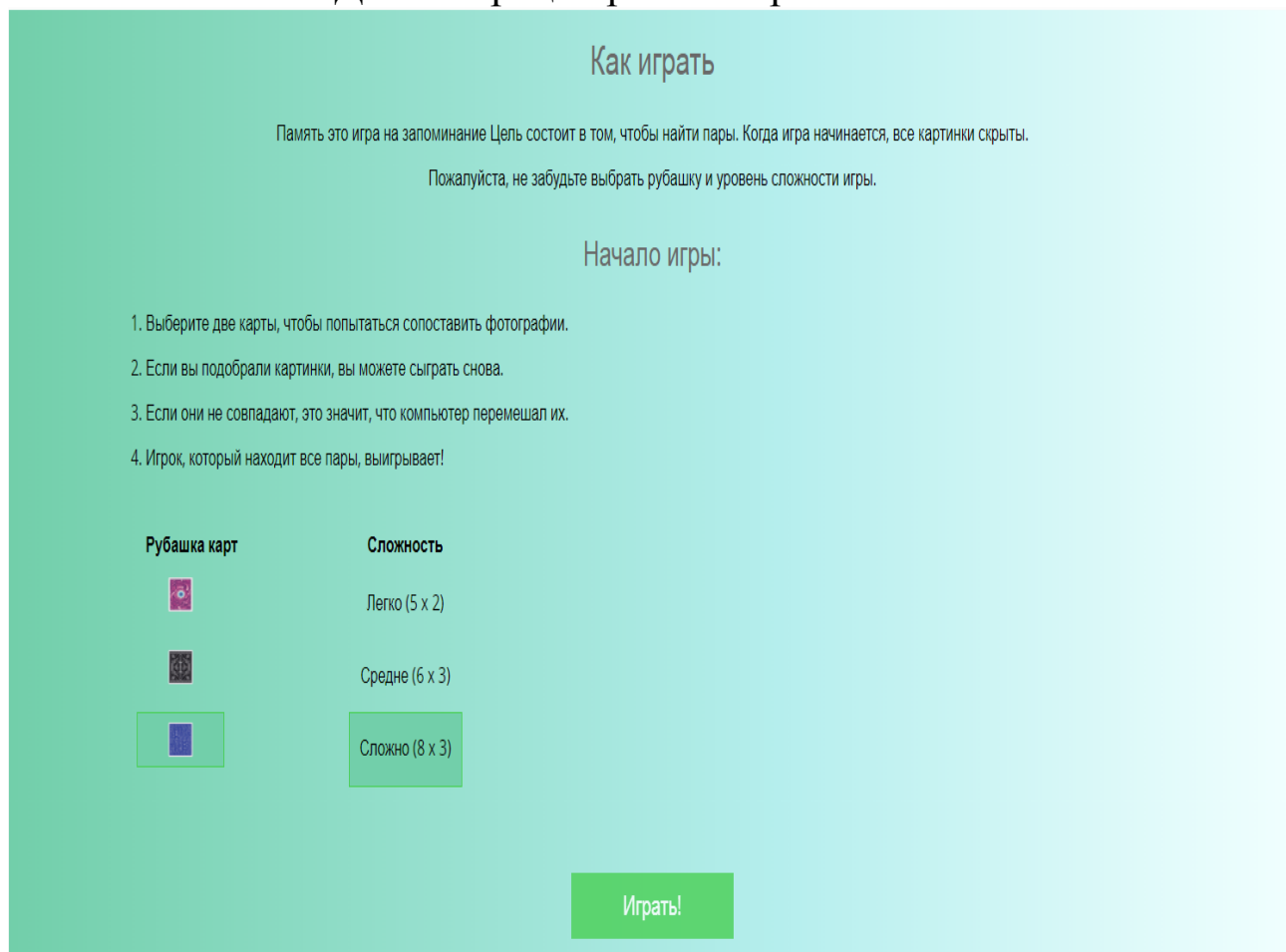


Рисунок А.1 – Общий вид интерфейса выбора рубашки и сложности

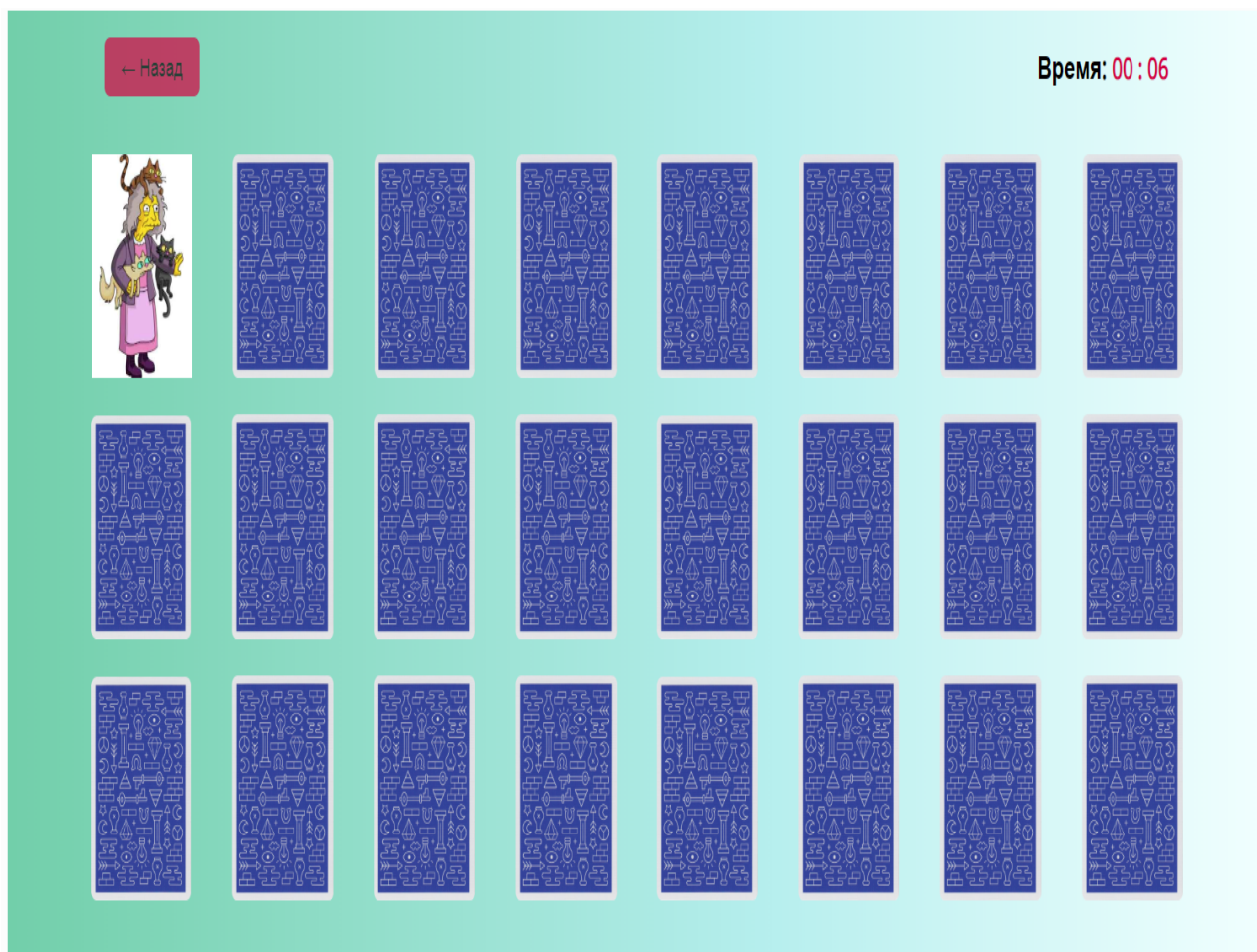


Рисунок А.2 – Начало игры, пользователю нужно найти 2 одинаковых карты.

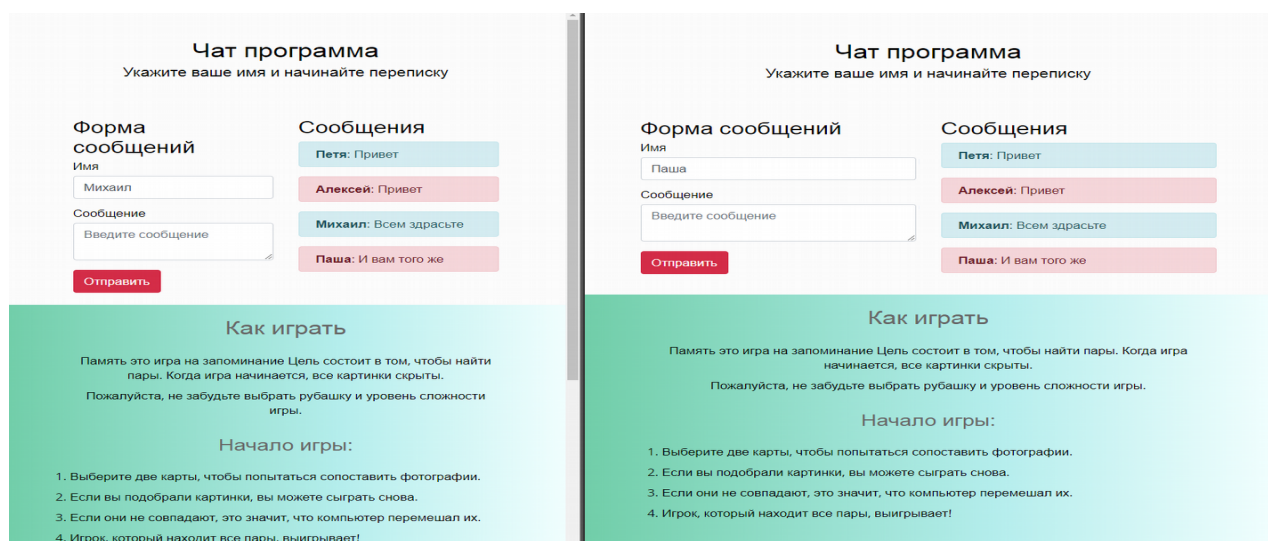


Рисунок А.3 – Демонстрация работы чата.

```
Microsoft Windows [Version 6.3.9600]  
(c) Корпорация Майкрософт (Microsoft Corporation), 2013. Все права защищены.  
  
C:\Users\Ярик\Desktop\Chat>node index  
Успешное соединение  
Отключились  
Успешное соединение  
Успешное соединение
```

Рисунок А.4 – Информация о подключении или отключении
пользователей

Приложение Б

(обязательное)

Листинг программы

Файл: game.js

```
'use strict';

const ENTER_KEYCODE = 13;
const ESC_KEYCODE = 27;
const rules = document.querySelector('.rules');
const form = document.querySelector('.form');
const btn = document.querySelector('.form__btn');
const formListShirt = document.querySelector('.form__list--shirt');
const formImage = document.querySelectorAll('.form__image');
const formListLevel = document.querySelector('.form__list--level');
const formItemLevel = document.querySelectorAll('.form__text--level');

const cards = document.querySelector('.cards');
const cardsBtn = document.querySelector('.cards__btn');
const cardsItems = document.querySelector('.cards__items');
const timeInSeconds = document.getElementById('sec');
const timeInMinutes = document.getElementById('min');
let countTime;

const popupGame = document.querySelector('.popup--game');
const popupBtnExit = document.querySelector('.popup__btn--exit');
const failure = document.querySelector('.popup--failure');
const popupBtnFailure = document.querySelector('.popup__btn--failure');
const popupTime = document.querySelector('.popup__title--time'); // поле вывода результата игроку
const popupBtnGame = document.querySelector('.popup__btn--game');
const popupTable = document.querySelector('.popup__table');

let firstTurnedCardIndex; // дефолтное значение индекса первой карты
let firstTurnedCardId; // дефолтное значение data-id первой карты

const player = { // объект с инфой игрока
  name: null,
  surname: null,
  email: null,
  score: null
};

let ratingList = []; // массив всех результатов
let ratingItem = []; // массив куда запишем имя и время

let memoryObj = {}; // запоминает какую выбрали рубашку и сложность для игры
```

```
let newArrCardsRandomAndSelected = []; // массив рандомных карт согласно опций
const dataOfCards = [{
  dataId: 1,
  backgroundImage: 'Barney_Gumble.jpg'
},
{
  dataId: 2,
  backgroundImage: 'Clancy_Wiggum.jpg'
},
{
  dataId: 3,
  backgroundImage: 'Bart_Simpson.jpg'
},
{
  dataId: 4,
  backgroundImage: 'Eleanor_Abernathy.jpg'
},
{
  dataId: 5,
  backgroundImage: 'Groundskeeper_Willie.jpg'
},
{
  dataId: 6,
  backgroundImage: 'Homer_Simpson.jpg'
},
{
  dataId: 7,
  backgroundImage: 'Horatio_McCallister.jpg'
},
{
  dataId: 8,
  backgroundImage: 'Fat_Tony.jpg'
},
{
  dataId: 9,
  backgroundImage: 'Mr_Burns.jpg'
},
{
  dataId: 10,
  backgroundImage: 'Moe_Szyslak.jpg'
},
{
  dataId: 11,
  backgroundImage: 'Abraham_Simpson.jpg'
},
{
  dataId: 12,
  backgroundImage: 'Seymour_Skinner.jpg'
}
];

// функция рандомного перемешивания
Array.prototype.shuffle = function () {
```

```

for (let i = this.length - 1; i >= 0; i--) {
  let randomIndex = Math.floor(Math.random() * (i + 1));
  let itemAtIndex = this[randomIndex];
  this[randomIndex] = this[i];
  this[i] = itemAtIndex;
}
return this;
}

```

// функция создания рандомного массива согласно выбранного уровня

```

function randomMixArrays(start, end) {
  let arrCut = dataOfCards.slice(start, end);
  let arrCopy = arrCut.slice();
  newArrCardsRandomAndSelected = arrCut.concat(arrCopy);
  newArrCardsRandomAndSelected.shuffle();
  return newArrCardsRandomAndSelected;
}

```

```

const fragment = document.createDocumentFragment();
const cardsItem = document.createElement('img');
fragment.appendChild(cardsItem);
cardsItem.classList.add('cards__item');

```

// функция начала игры

```

function init(obj) {
  // рандомное перемешивание первонач массива
  dataOfCards.shuffle();
  // поменять рубашку карт согласно наличию класса form__active
  for (let i = 0; i < formImage.length; i++) {
    if (formImage[i].classList.contains('form__active')) {
      obj.style.backgroundImage = "url('./img/" + i + ".png')";
      obj.classList.remove('cards__item--turned'); // у всех карт убрать классы переверачив
ания
      memoryObj.shirt = obj.style.backgroundImage; // сохраняем в св-во объекта инфу о в
ыбранной рубашке
    }
  }
  // выложить карты (+новый класс) согласно наличию класса form__active
  if (formItemLevel[1].classList.contains('form__active')) {
    randomMixArrays(0, 9); // добавить 18 карт
    addCards(); // добавление карт на поле
  } else if (formItemLevel[2].classList.contains('form__active')) {
    randomMixArrays(); // добавить 24 карты
    addCards();
  } else {
    randomMixArrays(0, 5); // добавить 10 карт
    addCards();
  }
  return memoryObj; // возврат объекта с инфой о выбранной рубашке и уровне сложно
сти
}

```

// функция отрисовки каждой карты


```

let renderCard = function (card, index) {
  const newCard = document.createElement('img');
  newCard.dataset.id = newArrCardsRandomAndSelected[index].dataId; // каждой карте д
аём id
  newCard.dataset.bg = newArrCardsRandomAndSelected[index].backgroundImage; // сохр
яняем картинку в атрибут data-bg
  newCard.style.backgroundImage = memoryObj.shirt;
  newCard.src = 'img/transparent.png';
  return newCard;
};

```

// функция добавления карт на поле

```

let addCards = function () {
  for (let i = 0; i < newArrCardsRandomAndSelected.length; i++) {
    let elem = renderCard(newArrCardsRandomAndSelected[i], i);
    elem.className = 'cards__item cards__item--medium-difficulty';
    if (newArrCardsRandomAndSelected.length < 18) {
      elem.className = 'cards__item cards__item--low-difficulty';
    }
    if (newArrCardsRandomAndSelected.length > 18) {
      elem.className = 'cards__item cards__item--hard-difficulty';
    }
    cardsItems.appendChild(elem); // добавляем карты на поле
  }
};

```

// таймер

```

function calcTime(sec, min, zeroing) { // при наличии 3-го парам-ра обнуляется таймер
  sec = Number(timeInSeconds.textContent);
  min = Number(timeInMinutes.textContent);
  sec++;
  if (zeroing) {
    sec = 0;
    min = 0;
  }
  if (sec >= 60) {
    sec = 0;
    min++;
  }
  if (sec < 10) {
    sec = '0' + sec;
  }
  if (min < 10) {
    min = '0' + min;
  }
  timeInSeconds.textContent = sec;
  timeInMinutes.textContent = min;
}

```

// функция выхода из игрового поля

```

function closeCardsField() {
  cards.style.display = 'none';
  rules.style.display = 'block';
}

```

```

form.style.display = 'block';
cardsItems.innerHTML = ""; // удаление карт
}

// функция вывода поздравлений
function outputResult() {
    clearInterval(countTime); // остановка таймера
    popupTime.textContent = min.textContent + ':' + sec.textContent; // вывод результата таймера
    if (!popupGame.classList.contains('popup--show')) {
        popupGame.classList.add('popup--show');
    }

    // заполняем профиль игрока в объект и добавляем в хранилище
    player.name = document.getElementsByName('name')[0].value;
    player.surname = document.getElementsByName('surname')[0].value;
    player.email = document.getElementsByName('email')[0].value;
    player.score = +timeInMinutes.textContent * 60 + +timeInSeconds.textContent; // перев
од в секунды
    window.localStorage.setItem('player', JSON.stringify(player)); // в хранилище будет доба
влено значение

    // добавляем время игрока в массив и сохраняем таблицу 10-ти лучших в хранилище
    ratingList = JSON.parse(window.localStorage.getItem('ratingList')); // вернёт массив знач
ений лежащих в хранилище
    if (!ratingList) {
        ratingList = [];
    }
    if (ratingList.length == 10) {
        ratingList.sort(function (a, b) {
            return a[1] - b[1];
        });
        ratingList = ratingList.slice(0, 9);
    }

    const nameUser = document.getElementsByName('name')[0].value;
    let scoreUser = +timeInMinutes.textContent * 60 + +timeInSeconds.textContent; // перев
од в секунды
    ratingItem[0] = nameUser;
    ratingItem[1] = scoreUser;
    ratingList.push(ratingItem);

    if ((ratingList.length > 0) && ratingList[ratingList.length - 1][1] > scoreUser) {
        ratingList[ratingList.length - 1][0] = nameUser;
        ratingList[ratingList.length - 1][1] = scoreUser;
    }
    if (ratingList.length > 1) {
        ratingList.sort(function (a, b) {
            return a[1] - b[1];
        });
    }
    window.localStorage.setItem('ratingList', JSON.stringify(ratingList)); // в хранилище будет
добавлено значение

```

```

// отрисовываем таблицу результатов
const ratingTable = document.createElement('table');
const headRow = document.createElement('tr');
ratingTable.appendChild(headRow);
ratingTable.classList.add('popup__table-tag');

// отрисовываем шапку
for (let i = 0; i < 3; i++) {
  const headCell = document.createElement('th');
  headCell.classList.add('popup__cell');
  if (i == 0) headCell.innerHTML = '№';
  if (i == 1) headCell.innerHTML = 'Name';
  if (i == 2) headCell.innerHTML = 'Time';
  headRow.appendChild(headCell);
}

for (let i = 0; i < 10; i++) {
  const tableRow = document.createElement('tr');
  ratingTable.appendChild(tableRow);
  for (let j = 0; j < 3; j++) {
    const tableCell = document.createElement('td');
    if (j == 0) tableCell.innerHTML = `${i + 1}`; // внести номер позиции
    if (ratingList[i]) {
      if (j == 1) tableCell.innerHTML = `${ratingList[i][0]}`; // внести имя
      if (j == 2) tableCell.innerHTML = `${ratingList[i][1]}`; // внести время
    }
    tableCell.classList.add('popup__cell');
    tableRow.appendChild(tableCell);
  }
}
popupTable.appendChild(ratingTable);
}

// обработчик поворота карты
cardsItems.addEventListener('click', function (e) {
  if (e.target.classList.contains('cards__item--turned') && !
e.target.classList.contains('cards__items')) {
    e.target.style.backgroundImage = memoryObj.shirt; // смена картинки на рубашку
    e.target.classList.toggle('cards__item--turned');
    firstTurnedCardId = null; // удаляем id первой карты из глобальной области видимост
и
    firstTurnedCardIndex = null; // удаляем индекс первой карты из глобальной области в
идимости

  } else if (!e.target.classList.contains('cards__items')) {
    e.target.classList.toggle('cards__item--turned');
    setTimeout(function () {
      e.target.style.backgroundImage = "url('./img/" + e.target.getAttribute('data-bg') + ""), u
rl('./img/white.png')";
    }, 300);
    const arrOfCards = document.querySelectorAll('.cards__item'); // массив карт
    let count = 0; // счётчик для кол-ва открытых карт

```

```

for (let i = 0; i < arrOfCards.length; i++) {
  if (arrOfCards[i].classList.contains('cards__item--turned')) {
    count++;
  }
  if (count == 1 && arrOfCards[i].classList.contains('cards__item--turned')) { // находим первую открытую карту и сохраняем её id
    if (!firstTurnedCardId) { // если индекса первой карты не задан, то задаём id первой карты
      firstTurnedCardId = arrOfCards[i].getAttribute('data-id');
    }
    if (!firstTurnedCardIndex) { // если индекса первой карты не задан, то задаём
      firstTurnedCardIndex = i;
    }
  }
  if (count == 2) { // если уже есть 2 открытые карты

    if (e.target.getAttribute('data-id') == firstTurnedCardId) { // сравниваем id активной карты и открытой, если они равны
      setTimeout(function () {
        e.target.style.visibility = 'hidden'; // скрываем вторую карту
        e.target.classList.remove('cards__item--turned');
        e.target.classList.remove('cards__item'); // убираем из массива
        arrOfCards[firstTurnedCardIndex].style.visibility = 'hidden'; // скрываем первую карту
        arrOfCards[firstTurnedCardIndex].classList.remove('cards__item--turned');
        arrOfCards[firstTurnedCardIndex].classList.remove('cards__item'); // убираем из массива
        firstTurnedCardId = null; // удаляем id первой карты
        firstTurnedCardIndex = null; // удаляем индекс первой карты из глобальной области видимости
      }, 500);
      if (arrOfCards.length < 4) { // вывод поздравлений игроку
        setTimeout(function () {
          outputResult();
        }, 700);
      }
    } else { // если id не совпали
      setTimeout(function () {
        e.target.style.backgroundColor = memoryObj.shirt; // закрываем вторую карту
        e.target.classList.toggle('cards__item--turned');
        arrOfCards[firstTurnedCardIndex].style.backgroundColor = memoryObj.shirt; // закрываем первую карту
        arrOfCards[firstTurnedCardIndex].classList.toggle('cards__item--turned');
        firstTurnedCardId = null; // удаляем id первой карты из глобальной области видимости
        firstTurnedCardIndex = null; // удаляем индекс первой карты из глобальной области видимости
      }, 700);
    }
    break;
  }
}

```

```
}  
});
```

```
// обработчики выделения выбранной опции  
formListShirt.addEventListener('click', function (e) {  
  for (let i = 0; i < formImage.length; i++) {  
    formImage[i].classList.remove('form__active');  
  }  
  e.target.classList.toggle('form__active');  
  formListShirt.classList.remove('form__active');  
});
```

```
formListLevel.addEventListener('click', function (e) {  
  for (let i = 0; i < formItemLevel.length; i++) {  
    formItemLevel[i].classList.remove('form__active');  
  }  
  e.target.classList.add('form__active');  
  formListLevel.classList.remove('form__active');  
});
```

```
// валидация на вход в игру  
btn.addEventListener('click', function (event) {  
  event.preventDefault();  
  rules.style.display = 'none';  
  form.style.display = 'none';  
  cards.style.display = 'block';  
  init(cardsItem); // инициализация игры  
  countTime = setInterval(calcTime, 1000); // запуск таймера  
  
});
```

```
// обработчики досрочного выхода из игрового поля  
let cardsBtnClickHandler = function () {  
  clearInterval(countTime); // остановка таймера  
  calcTime(0, 0, 1); // обнуление таймера  
  closeCardsField(); // выход из игрового поля  
  cards.removeEventListener('click', cardsBtnClickHandler);  
}
```

```
let enterPressHandler = function (event) {  
  clearInterval(countTime);  
  if (event.keyCode === ENTER_KEYCODE) {  
    closeCardsField();  
    cards.removeEventListener('keydown', enterPressHandler);  
  }  
}
```

```
cardsBtn.addEventListener('click', cardsBtnClickHandler);  
cardsBtn.addEventListener('keydown', enterPressHandler);
```

```
// обработчик закрытия попапа  
popupBtnFailure.addEventListener('click', function () {  
  if (failure.classList.contains('popup--show')) {
```

```

        failure.classList.remove('popup--show');
    }
});

// обработчик закрытия попапов ESC
window.addEventListener('keydown', function (event) {
    if (event.keyCode === ESC_KEYCODE) {
        if (popupGame.classList.contains('popup--show')) {
            popupGame.classList.remove('popup--show');
            clearInterval(countTime); // остановка таймера
            closeCardsField(); // выход из игрового поля
        }
        if (failure.classList.contains('popup--show')) {
            failure.classList.remove('popup--show');
        }
    }
});

// обработчик для новой игры
popupBtnGame.addEventListener('click', function (event) {
    if (popupGame.classList.contains('popup--show')) {
        popupGame.classList.remove('popup--show');
    }
    cardsItems.innerHTML = ""; // удаление карт
    calcTime(0, 0, 1); // обнуление таймера
    init(cardsItem); // инициализация игры
    countTime = setInterval(calcTime, 1000);
});

// обработчик для выхода из игры
popupBtnExit.addEventListener('click', function () {
    if (popupGame.classList.contains('popup--show')) {
        popupGame.classList.remove('popup--show');
    }
    calcTime(0, 0, 1); // обнуление таймера
    closeCardsField(); // выход из игрового поля
});

```

Файл: game.js

<!DOCTYPE html

|>

```

<html lang="ru">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <!-- Подключение Bootstrap чтобы все выглядело красиво -->
    <link

```

```

    rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css"
  />
  <link href="/css/style.css" rel="stylesheet" />
  <title>ChatGame</title>
  <!-- Свои стили -->
  <style>
    body {
      background: #fcfcfc;
    }
  </style>
</head>

<body>
  <!-------CHAT----->
  <!-- Основная часть страницы -->
  <div class="container">
    <div class="py-5 text-center">
      <h2>Чат программа</h2>
      <p class="lead">Укажите ваше имя и начинайте переписку</p>
    </div>
    <div class="row">
      <div class="col-6">
        <!-- Форма для получения сообщений и имени -->
        <h3>Форма сообщений</h3>
        <form id="messForm">
          <label for="name">Имя</label>
          <input
            type="text"
            name="name"
            id="name"
            placeholder="Введите имя"
            class="form-control"
          />
          <br />
          <label for="message">Сообщение</label>
          <textarea
            name="message"
            id="message"
            class="form-control"
            placeholder="Введите сообщение"
          ></textarea>
          <br />
          <input type="submit" value="Отправить" class="btn btn-danger" />
        </form>
      </div>
      <div class="col-6">
        <h3>Сообщения</h3>
        <!-- Вывод всех сообщений будет здесь -->
        <div id="all_mess"></div>
      </div>
    </div>
  </div>
</body>

```



```

<!-- Подключаем jQuery, а также Socket.io -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
>
<script src="/socket.io/socket.io.js"></script>
<script>
    // У каждого пользователя будет случайный стиль для блока с сообщениями,
    // поэтому в этом кусочке кода мы получаем случайные числа
    var min = 1;
    var max = 6;
    var random = Math.floor(Math.random() * (max - min)) + min;

    // Устанавливаем класс в переменную в зависимости от случайного числа
    // Эти классы взяты из Bootstrap стилей
    var alertClass;
    switch (random) {
        case 1:
            alertClass = "secondary";
            break;
        case 2:
            alertClass = "danger";
            break;
        case 3:
            alertClass = "success";
            break;
        case 4:
            alertClass = "warning";
            break;
        case 5:
            alertClass = "info";
            break;
        case 6:
            alertClass = "light";
            break;
    }

    // Функция для работы с данными на сайте
    $(function() {
        // Включаем socket.io и отслеживаем все подключения
        var socket = io.connect();
        // Делаем переменные на:
        var $form = $("#messForm"); // Форму сообщений
        var $name = $("#name"); // Поле с именем
        var $textarea = $("#message"); // Текстовое поле
        var $all_messages = $("#all_mess"); // Блок с сообщениями

        // Отслеживаем нажатие на кнопку в форме сообщений
        $form.submit(function(event) {
            // Предотвращаем классическое поведение формы
            event.preventDefault();
            // В сокет отсылаем новое событие 'send mess',
            // в событие передаем различные параметры и данные
            socket.emit("send mess", {
                mess: $textarea.val(),
            });
        });
    });

```

```

        name: $name.val(),
        className: alertClass
    });
    // Очищаем поле с сообщением
    $textarea.val("");
    });

    // Здесь отслеживаем событие 'add mess',
    // которое должно приходить из сокета в случае добавления нового сообщен
ия
    socket.on("add mess", function(data) {
        // Встраиваем полученное сообщение в блок с сообщениями
        // У блока с сообщением будет тот класс, который соответствует пользовател
ю что его отправил
        $all_messages.append(
            "<div class='alert alert-" +
                data.className +
                "'><b>" +
                data.name +
                "</b>: " +
                data.mess +
                "</div>"
        );
    });
});
</script>
<!-------GAME----->
<section class="rules">
    <div class="rules__wrapper">
        <h1 class="rules__title">Как играть</h1>
        <p class="rules__text rules__text--general">
            Память это игра на запоминание Цель состоит в том, чтобы найти пары.
            Когда игра начинается, все картинки скрыты.
        </p>
        <p class="rules__text rules__text--general">
            Пожалуйста, не забудьте выбрать рубашку и уровень сложности игры.
        </p>
        <h2 class="rules__title rules__title--play">Начало игры:</h2>
        <ol class="rules__text-container">
            <li class="rules__text">
                Выберите две карты, чтобы попытаться сопоставить фотографии.
            </li>
            <li class="rules__text">
                Если вы выбрали картинки, вы можете сыграть снова.
            </li>
            <li class="rules__text">
                Если они не совпадают, это значит, что компьютер перемешал их.
            </li>
            <li class="rules__text">
                Игрок, который находит все пары, выигрывает!
            </li>
        </ol>
    </div>

```

```
</section>
```

```
<section class="form">
  <div class="form__wrapper">
    <div class="form__option-container">
      <div class="form__option">
        <p class="form__title form__title--option">Рубашка карт</p>
        <ul class="form__list form__list--shirt">
          <li class="form__text">
            
          </li>
          <li class="form__text">
            
          </li>
          <li class="form__text">
            
          </li>
        </ul>
      </div>

      <div class="form__option">
        <p class="form__title form__title--option">Сложность</p>
        <ul class="form__list form__list--level">
          <li class="form__text form__text--level form__active">
            Легко (5 x 2)
          </li>
          <li class="form__text form__text--level">Средне (6 x 3)</li>
          <li class="form__text form__text--level">Сложно (8 x 3)</li>
        </ul>
      </div>
    </div>

    <div class="form__btn-container">
      <button class="form__btn">Играть!</button>
    </div>
  </div>
</section>
```

```
<section class="cards">
  <div class="cards__wrapper">
    <div class="cards__nav-container">
      <button class="cards__btn" type="submit">← Назад</button>
    <p class="cards__text">
      Время:
      <span class="cards__timer">
        <span id="min">00</span> : <span id="sec">00</span>
      </span>
    </p>
  </div>
</section>
```

```

        <div class="cards__items"></div>
    </div>
</section>

<section class="popup popup--game">
    <div class="popup_wrapper">
        <h3 class="popup_title">Поздравляем!</h3>
        <h3 class="popup_title">Ваше время:</h3>
        <p class="popup_title popup_title--time"></p>
        <div class="popup_table"></div>
        <button class="popup_btn popup_btn--game" type="submit">
            Новая игра
        </button>
        <button class="popup_btn popup_btn--exit" type="submit">
            Выйти
        </button>
    </div>
</section>

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

```

Файл style.css

```

* {

    margin: 0;
    padding: 0;
}

body {
    width: 100%;
    height: 100%;
    line-height: 1;
    font-family: "Open sans", Arial, sans-serif;
    font-size: 16px;
    font-weight: 400;
    color: #000;
    background-image: linear-gradient(to right, #A9A9A9, #D3D3D3, #F5F5F5);
}

.cards {
    position: relative;
    display: none;
    width: 100%;
    max-width: 1500px;
    margin: 0 auto;
    margin-top: 20px;
    margin-bottom: 20px;
}

```

```
background-image: linear-gradient(to right, #66CDAA 0%, #AFEEEE 60%, #F0FFFF 100%);
z-index: 1;
}
```

```
.cards__wrapper {
width: 90%;
margin: 0 auto;
padding: 20px 0 30px;
}
```

```
.cards__nav-container {
display: flex;
justify-content: space-between;
margin-bottom: 30px;
padding: 0 3%;
}
```

```
.cards__btn {
display: block;
padding: 10px 20px;
text-align: center;
border: none;
font-size: 18px;
font-weight: 400;
background-color: #DC143C;
color: #000;
border-radius: 7px;
text-decoration: none;
cursor: pointer;
overflow: hidden;
filter: opacity(70%);
}
```

```
.cards__btn:hover {
filter: opacity(85%);
}
```

```
.cards__btn:active {
filter: opacity(100%);
}
```

```
.cards__text {
text-align: right;
margin: 10px 0;
font-size: 24px;
font-weight: 700;
}
```

```
.cards__timer { /* таймер */
font-size: 22px;
color: #DC143C;
}
```

```

.cards__items {
  width: 100%;
  margin: 0 auto;
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
}

.cards__item {
  margin: 1% 0;
  cursor: pointer;
  background-repeat: no-repeat;
  background-size: contain;
  background-position: center;
}

.cards__item--low-difficulty { /* 5x2 */
  width: 20%;
  height: 20%;
}

.cards__item--medium-difficulty { /* 6x3 */
  width: 16.66%;
  height: 16.66%;
}

.cards__item--hard-difficulty { /* 8x3 */
  width: 12.5%;
  height: 12.5%;
}

@keyframes rotate {
  0% {
    transform: rotate3d(0, 1, 0, 0deg);
  }
  50% {
    transform: rotate3d(0, 1, 0, 90deg);
  }
  100% {
    transform: rotate3d(0, 1, 0, 0deg);
  }
}

.cards__item--turned {
  animation-name: rotate;
  animation-duration: 0.5s;
  animation-fill-mode: forwards;
}

@font-face {
  font-family: 'Open sans';
  src: url('../fonts/opensans-regular-webfont.woff2') format('woff2'),
       url('../fonts/opensans-regular-webfont.woff') format('woff');
}

```

```
font-weight: normal;
font-style: normal;
}
.form {
  position: relative;
  width: 100%;
  max-width: 1500px;
  margin: 0 auto;
  margin-bottom: 20px;
  background-image: linear-gradient(to right, #66CDAA 0%, #AFEEEE 60%, #F0FFFF 100
%);
}

.form__wrapper {
  width: 80%;
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
  margin: 0 auto;
  padding-bottom: 30px;
}

.form__form-container {
  width: 360px;
}

.form__items {
  width: 100%;
  border: 0;
  text-align: right;
}

.form__item {
  display: block;
  width: 100%;
  margin: 25px 0;
}

.form__title {
  margin: 0 10px;
  font-weight: 700;
}

.form__title--option {
  text-align: center;
  margin-bottom: 10px;
}

.form__input {
  width: 230px;
  height: 36px;
  padding: 10px;
  text-align: left;
```



```
border: 1px solid #c8c6c6;
outline: 0;
box-sizing: border-box;
overflow: hidden;
}

.form__input:focus {
border-color: #32CD52;
}

.form__option-container {
display: flex;
justify-content: space-around;
flex-direction: row;
width: 50%;
min-width: 400px;
}

.form__option:last-child {
margin-right: auto;
margin-left: 20%;
}

.form__text {
display: block;
width: 100px;
height: 55px;
text-align: center;
outline: 0;
box-sizing: border-box;
}

.form__text:hover {
outline: 1px solid #32CD52;
}

.form__text:active {
background-color: #66CDAA;
}

.form__text--level {
width: 130px;
padding: 18px 0;
}

.form__active {
/* при срабатывании обработчика событий */
background-color: #66CDAA;
outline: 1px solid #32CD52;
}

.form__image {
width: 100px;
```

```
height: 40px;
padding: 7px 30px;
}
```

```
.form__btn-container {
width: 100%;
margin: 0 auto;
padding-top: 50px;
}
```

```
.form__btn {
display: block;
margin: 0 auto;
padding: 15px 60px;
text-align: center;
color: #fff;
font-size: 20px;
cursor: pointer;
background-color: #32CD52;
border: none;
overflow: hidden;
outline: none;
filter: opacity(80%);
}
```

```
.form__btn:hover {
filter: opacity(100%);
}
```

```
.form__btn:active {
background-color: #228B22;
}.popup { /* в скрытом состоянии */
display: none;
position: fixed;
width: 460px;
left: 50%;
top: 3%;
transform: translate(-50%, -50%);
z-index: 5;
}
```

```
.popup--show { /* Отобразит popup */
display: block;
}
```

```
.popup__wrapper {
position: absolute;
width: 100%;
padding: 20px 50px 5px;
background-color: white;
box-shadow: 0 5px 20px 0 #32CD52;
box-sizing: border-box;
}
```

```
.popup__wrapper--failure {  
  box-shadow: 0 5px 20px 0 #DC143C;  
}
```

```
.popup__title {  
  margin-top: 14px;  
  text-align: center;  
  font-size: 30px;  
  font-weight: 700;  
}
```

```
.popup__title--time {  
  color: #dc143c;  
  margin-bottom: 30px;  
}
```

```
.popup__table {  
  max-width: 358px;  
  margin: 0 auto;  
  margin-bottom: 30px;  
}
```

```
.popup__table-tag {  
  max-width: 358px;  
  margin: 0 auto;  
  border-collapse: collapse;  
}
```

```
.popup__cell {  
  border: 1px solid #000;  
  width: auto;  
  min-width: 40px;  
  max-width: 268px;  
  overflow: hidden;  
  padding: 3px 10px;  
  text-overflow: ellipsis;  
  box-sizing: border-box;  
}
```

```
.popup__text {  
  font-size: 30px;  
  text-align: center;  
  line-height: 24px;  
  margin: 20px 0 40px;  
}
```

```
.popup__text--failure {  
  font-size: 24px;  
}
```

```
.popup__btn {  
  display: block;
```

```
width: 250px;
margin: 0 auto;
margin-bottom: 30px;
padding: 15px 40px;
text-align: center;
border: none;
font-size: 24px;
font-weight: 700;
background-color: #DC143C;
color: white;
text-decoration: none;
cursor: pointer;
overflow: hidden;
filter: opacity(70%);
}

.popup__btn:hover,
.popup__btn--game:hover {
  filter: opacity(85%);
}

.popup__btn:active,
.popup__btn--game:active {
  filter: opacity(100%);
}

.popup__btn--game {
  background-color: #32CD52;
}

.rules {
  position: relative;
  width: 100%;
  max-width: 1500px;
  margin: 0 auto;
  margin-top: 20px;
  background-image: linear-gradient(to right, #66CDAA 0%, #AFEEEE 60%, #FFFFFF 100%);
}

.rules__wrapper {
  width: 80%;
  margin: 0 auto;
  padding: 20px 0 30px;
}

.rules__title {
  margin-bottom: 25px;
  text-align: center;
  font-size: 30px;
  color: #696969;
```

```
}
```

```
.rules__title--play {  
  margin-top: 30px;  
  font-size: 26px;  
}
```

```
.rules__text-container {  
  width: 100%;  
  max-width: 1080px;
```

```
}
```

```
.rules__text {  
  margin: 10px;  
  font-size: 16px;  
  text-align: left;  
  line-height: 1.5;  
}
```

```
.rules__text--general {  
  text-align: center;  
}
```

```
@import "base.css";  
@import "fonts.css";  
@import "rules.css";  
@import "form.css";  
@import "cards.css";  
@import "popup.css";
```