

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО» ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

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

Лабораторна робота №1

з дисципліни «Web-дизайн»

Тема: «Розробка функціональності веб додатка мовою JavaScript»

Виконав: студент III курсу
ФПМ групи KB-83
Кубай О. Ф.
Викладач: Петрашенко А. В.

Київ 2020

Мета: ознайомитись із засобами мови Javascript та навчитись їх застосовувати побудови Web-інтерфейсу користувача.

Загальне завдання: розробити функціональність для статичних сторінок Web-додатку першої лабораторної роботи із використанням шаблону MVC

Інструменти розробки: мови HTML5, CSS3, Javascript.

Зауваження: Web-фреймворки (ReactJS, AngularJS, VueJS) не використовувати!

Порядок виконання роботи:

- 1) Запропонувати тематику Web-додатку (див. рекомендований список).
- 2) Ознайомившись з переліком обов'язкових сторінок, виконати їх програмування:
 - a. Встановити середовище розробки (рекомендовано VS Code)
 - b. Ознайомитись з методичними вказівками щодо реалізації шаблону MVC
 - c. Виконати кодування додатку згідно із загальноприйнятими вимогами (див. <https://github.com/airbnb/javascript>).
 - d. Організувати код у вигляді модулів (класів) шаблону MVC (див. приклад додатку ToDo: <https://github.com/Aroxed/js-todo-mvc>).
 - e. Як сховище даних використовувати структури даних Javascript: масиви, об'єкти, рядки тощо.
- 4) За бажанням студента доповнити сайт іншими сторінками.
- 5) Створити репозиторій та завантажити усі файли на github.com.

Посилання на репозиторій: https://github.com/kubayof/web_labs

Контакт в Telegram: @kubayof (+380982755357)

Додаток завантажень на [github-pages](#), посилання можна знайти в README.md лабораторної роботи.

The diagram illustrates the architecture of a music application, organized into three main layers: Controller, Model, and View.

Controller Layer:

- Controller**: A base class with methods `viewFacade`, `modelFacade`, `registerListeners()`, and `setCallbacks()`. It has a composition relationship with **viewFacade** and **modelFacade**.
- ModelFacade**: A class that manages the model layer, containing methods like `chordsModelList`, `userPrincipalService`, `defaultInit()`, `getChordsModelList()`, `addChords()`, `removeChordsById()`, `signin()`, `signup()`, `signOut()`, and `isAuthorized()`. It has a composition relationship with **ChordsModelList** and **UserPrincipalService**.
- ChordsModelList**: A class that manages the list of chords, containing methods `items`, `add()`, and `remove()`. It has a composition relationship with **Chords**.
- Chords**: A class representing a chord, containing attributes `id`, `userId`, `title`, `musician`, `date`, and `text`. It has a composition relationship with **Comments**.
- Comments**: A class representing a comment, containing methods `comments` and `addComment()`. It has a composition relationship with **UserPrincipal**.
- UserPrincipalService**: A class that manages user principals, containing methods `userPrincipal`, `isAuthorized()`, `signin()`, `signup()`, and `signOut()`. It has a composition relationship with **UserPrincipal**.
- UserPrincipal**: A class representing a user principal, containing attributes `id`, `name`, `email`, `gender`, and `birthDate`.

View Layer:

- ViewFacade**: A base class for the view layer, containing methods like `contentWrapperView`, `navBarView`, `viewRegistry`, `contentView`, `registerListeners()`, `setContentViewName()`, `setContentView()`, `repaint()`, `rootElementName()`, `toHtml()`, and `innerHTML()`. It has a composition relationship with **ContentWrapperView**, **NavBarView**, **HomeView**, **WorkspaceView**, **MyAccountView**, **AboutView**, **SignInView**, **SignUpView**, and **FullChordsView**.
- ContentWrapperView**: A class that wraps the content, containing methods `jumbotronView`, `navBarView`, `viewRegistry`, `contentView`, `registerListeners()`, `setContentViewName()`, `setContentView()`, `repaint()`, `rootElementName()`, `toHtml()`, and `innerHTML()`. It has a composition relationship with **NavBarView**.
- NavBarView**: A class that manages the navigation bar, containing methods `viewFacade`, `activeTab`, `registerListeners()`, `onSignInOut()`, `repaint()`, `rootElementName()`, `toHtml()`, `innerHTML()`, `navBarEntriesHtml()`, `isActive()`, `userNavBarEntriesHtml()`, `authorizedDisplayStyle()`, and `nonAuthorizedDisplayStyle()`. It has a composition relationship with **ChordsModelListView**.
- ChordsModelListView**: A class that manages the list of chords, containing methods `chordsModelList`, `chordsView`, `registerListeners()`, and `toHtml()`. It has a composition relationship with **ChordsView**.
- ChordsView**: A class representing a chord view, containing methods `viewFacade`, `chords`, `registerListeners()`, `viewLinkId()`, `editLinkId()`, `deleteLinkId()`, `editButton()`, and `deleteButton()`. It has a composition relationship with **CommentsView**.
- CommentsView**: A class representing a comment view, containing methods `viewFacade`, `chords`, `registerListeners()`, `onAddComment()`, `repaint()`, `toHtml()`, `innerHTML()`, and `addCommentHtml()`. It has a composition relationship with **CommentsModelListView**.
- CommentsModelListView**: A class that manages the list of comments, containing methods `commentsModelList` and `commentsModelList`. It has a composition relationship with **CommentView**.
- CommentView**: A class representing a comment view, containing methods `comment` and `toHtml()`.
- HomeView**: A class that manages the home view, containing methods `viewFacade`, `chordsModelListView`, `registerListeners()`, and `toHtml()`. It has a composition relationship with **WorkspaceView**.
- WorkspaceView**: A class that manages the workspace view, containing methods `viewFacade`, `chords`, `registerListeners()`, `setChords()`, `onAddChords()`, `toHtml()`, `signin()`, `signup()`, `signOut()`, `isAuthorized()`, `signin()`, `signup()`, and `signOut()`. It has a composition relationship with **MyAccountView**.
- MyAccountView**: A class that manages the my account view, containing methods `userPrincipalView`, `registerListeners()`, and `toHtml()`. It has a composition relationship with **AboutView**.
- AboutView**: A class that manages the about view, containing methods `registerListeners()` and `toHtml()`. It has a composition relationship with **SignInView**.
- SignInView**: A class that manages the sign in view, containing methods `viewFacade`, `registerListeners()`, `onSignIn()`, and `toHtml()`. It has a composition relationship with **SignUpView**.
- SignUpView**: A class that manages the sign up view, containing methods `viewFacade`, `registerListeners()`, `onSignUp()`, and `toHtml()`. It has a composition relationship with **FullChordsView**.
- FullChordsView**: A class that manages the full chords view, containing methods `chords`, `commentsView`, `registerListeners()`, `toHtml()`, `innerHTML()`, `isChord()`, and `commentsHtml()`. It has a composition relationship with **CommentsView**.

Скріншоти сторінок:

Home (signed in):



| | | |
|---|---|--|
| Jenny by Studio Killers View Edit 5/3/2021 | Jenny by Studio Killers View Edit 5/3/2021 | Jenny by Studio Killers View Edit 5/3/2021 |
| Jenny by Studio Killers View Edit 5/3/2021 | Jenny by Studio Killers View Edit 5/3/2021 | Jenny by Studio Killers View Edit 5/3/2021 |
| Jenny by Studio Killers View Edit 5/3/2021 | Jenny by Studio Killers View Edit 5/3/2021 | Jenny (2) by Studio Killers View Edit Delete 5/3/2021 |

[Back to top](#)

Home (signed out):

Jenny by Studio Killers

View

5/3/2021

Jenny by Studio Killers

View

5/3/2021

Jenny by Studio Killers

View

5/3/2021

Jenny by Studio Killers

View

5/3/2021

Jenny by Studio Killers

View

5/3/2021

Jenny by Studio Killers

View

5/3/2021

Jenny by Studio Killers

View

5/3/2021

Jenny (2) by Studio Killers

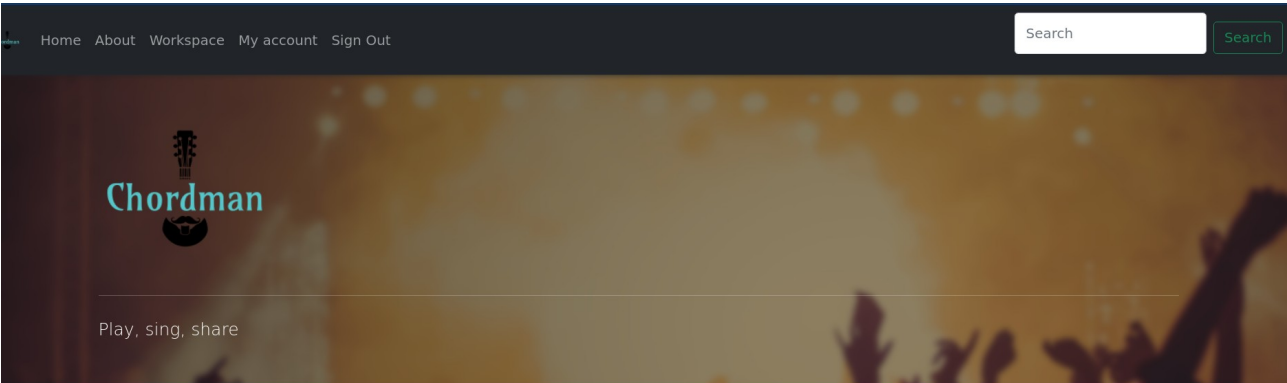
View

5/3/2021

[Back to top](#)

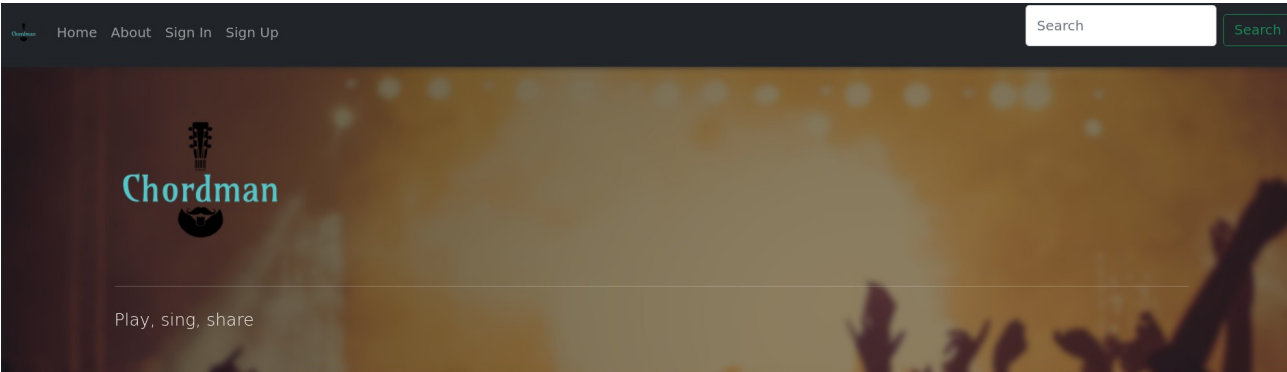


Navbar (signed in):



| | | |
|--|--|--|
| Jenny by Studio Killers View Edit 5/3/2021 | Jenny by Studio Killers View Edit 5/3/2021 | Jenny by Studio Killers View Edit 5/3/2021 |
| Jenny by Studio Killers View Edit 5/3/2021 | Jenny by Studio Killers View Edit 5/3/2021 | Jenny by Studio Killers View Edit 5/3/2021 |
| Jenny by Studio Killers | Jenny by Studio Killers | Jenny (2) by Studio Killers |

Navbar (Signed out):



| | | |
|--|--|--|
| Jenny by Studio Killers View Edit 5/3/2021 | Jenny by Studio Killers View Edit 5/3/2021 | Jenny by Studio Killers View Edit 5/3/2021 |
| Jenny by Studio Killers View Edit 5/3/2021 | Jenny by Studio Killers View Edit 5/3/2021 | Jenny by Studio Killers View Edit 5/3/2021 |


Workspace:

Chordman

HomeAboutWorkspaceMy accountSign Out

Search

Search



Play, sing, share

Example:
<Am> Hello <Cm> It's me

Name

Author

Lyrics and chords:


About:

Chordman

HomeAboutWorkspaceMy accountSign Out

Search

Search

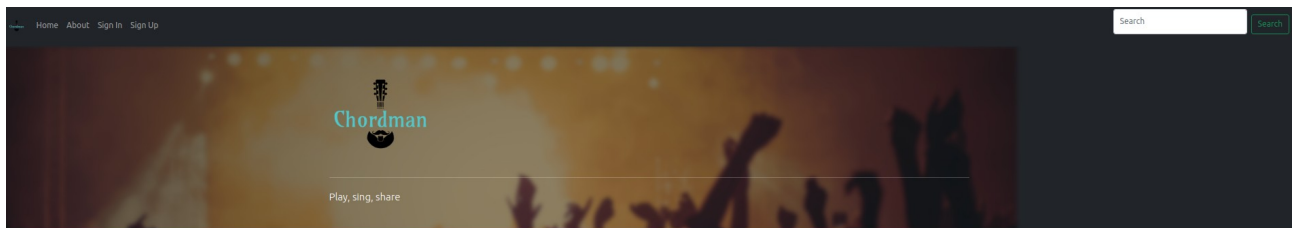



Play, sing, share

I really don't know what to write here, so:
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

[Back to top](#)

Sign up:





Chordman

Sign up

Name

Email

Password

Gender

Female

Birth date

mm/dd/yyyy

Sign up

Вміст JavaScript файлу робочої сторінки:

```
export default class WorkspaceView {
  constructor(viewFacade, chords = null) {
    this.viewFacade = viewFacade;
    this.chords = chords;
  }

  registerListeners() {
    document.getElementById('workspace_form').addEventListener('submit', e =>
this.onAddChords(e));
    this.setChords();
  }

  setChords() {
    if (this.chords != null) {
      document.getElementById('song_name').setAttribute('value', this.chords.title);
      document.getElementById('song_author').setAttribute('value', this.chords.musician);
      document.getElementById('song_text').defaultValue = this.chords.text;
    }
  }

  onAddChords(e) {
    e.preventDefault();
    const formData = new FormData(document.getElementById('workspace_form'));
    const name = formData.get('name');
    const author = formData.get('author');
    const text = formData.get('text');
    const date = new Date();
    this.viewFacade.workspaceOnAddChordsCallback(name, author, date, text);
  }

  toHtml() {
    return `
    <p>
      Example: <br/>
      &lt;Am&gt; Hello &lt;Cm&gt; It's me
    </p>
    <form id="workspace_form">
      <div class="form-group">
        <label for="exampleInputEmail1">Name</label>
        <input type="text" name="name" class="form-control" id="song_name" aria-
describedby="emailHelp"
          placeholder="Enter song's name">
        <label for="exampleInputEmail1">Author</label>
        <input type="text" name="author" class="form-control" id="song_author" aria-
describedby="emailHelp"
          placeholder="Author">
      </div>
      <div class="form-group">
        <label for="exampleFormControlTextarea1">Lyrics and chords:</label>
        <textarea name="text" class="form-control" id="song_text"
rows="50"></textarea>
      </div>
      <button type="submit" id="publish_button" class="btn btn-
primary">Publish</button>
    </form>
  `;
  }
}
```