НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КПІ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

**КУРСОВА РОБОТА**

***з дисципліни "Основи web-програмування"***

Виконала: Антушева Марія Ярославівна

Група: КП-73

Допущено до захисту

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1 семестр 2018/2019

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КПІ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

|  |  |
| --- | --- |
| Узгоджено    Керівник роботи    \_\_\_\_\_\_\_\_\_\_/Гадиняк Р.А./ | ЗАХИЩЕНА "\_\_"\_\_\_\_\_\_\_\_\_2017р.  з оцінкою\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_/Гадиняк Р.А./ |

Збірник гітарної табулатури і акордів

Виконавець роботи

Антушева Марія Ярославівна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_2018р.

**Вступ**

RockChords - це навчальний сайт з найбільшою колекцією гітарних акордів і табулатур для початківців та професіоналів, для тих, хто просто любить музику.

Користувачі можуть використовувати бібліотеку гітарних акордів (здійснювати їх пошук та відображення), переглядати бібліотеку табулатур із можливістю її відтворення. Вони також отримують онлайн-доступ до збірника, можуть здійснювати пошук колекцій за назвою або автором, створювати власні колекції акордів.

За допомогою RockChords та завдяки покроковим інструкціям, будь-хто може навчитися грати на гітарі, незалежно від рівня підготовки та віку, або інших несподіванок. На схемах акордів детально зображено, як брати акорди на грифі шестиструнної гітари. На всіх картинках найтонша струна розташована зверху, найтовстіша - знизу. Цифри під картинкою вказують лад, цифри справа - номер пальця, який затискає струну. Точки - розміщення пальців на грифі гітари. Якщо кілька точок знаходиться один під одним - це означає, що використовується прийом баре.

Тобто, дана курсова робота призначена для того, щоб мати швидкий онлайн-доступ до бібліотеки акордів та колекцій.

Отже, Rock Chords найбільш прогресивний навчальний сайт, що чудово підходить для самостійного вивчення акордів і табулатур.

**Інструменти розробки**

На основі [технічного завдання](about:blank) на виконання даної роботи було узгоджено створення веб-додатку у мережі Інтернет із використанням фреймворку NodeJS на мові програмування JavaScript. У якості сховища даних було обрано документну базу даних MongoDB. Для завантаження та збереження файлів було використано Cloudinary.

***JavaScript*  -** динамічна, [об'єктно-орієнтован](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)a [прототипна](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [мова програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F). Реалізація стандарту [ECMAScript](https://uk.wikipedia.org/wiki/ECMAScript). Найчастіше використовується для створення сценаріїв [веб-сторінок](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0), що надає можливість на стороні [клієнта](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0) (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, [асинхронно](https://uk.wikipedia.org/wiki/AJAX) обмінюватися даними з [сервером](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0), змінювати [структуру](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0) та [зовнішній вигляд](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD) [веб-сторінки](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0).

***NodeJS* -**це асинхронне середовище виконання JavaScript коду на моделі подій. Node.js призначений для побудови масштабованих мережевих додатків.

***MongoDB*  -** це документна база даних. Записи у MongoDB є документами, структура яких складається із пар ключ-значення. Такі документи дуже схожі із JSON-об'єктами. MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій. У MongoDB є вбудовані засоби із забезпечення шардінгу (розподіл набору даних по серверах на основі певного ключа), комбінуючи який реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу. Розширення кластера або перетворення одного сервера на кластер проводиться без зупинки роботи БД простим додаванням нових машин.

***Cloudinary –*** це сервіс, який може стати хмаровим сховищем для зображень та відео, може їх зжимати, конвертувати, редагувати. Навіть додавати до них різноманітні ефекти. Редагування відбувається миттєво, в особистому кабінеті без перезавантаження сторінки. Для роботи з сервісом необхідно пройти просту реєстрацію.

**Реалізований функціонал системи**

Відповідно до технічного завдання були враховані і реалізовані всі функціональні вимоги до системи:

1. Система надає користувачам швидкий спосіб реєстрації та авторизації у системі.

|  |
| --- |
| router.post('/register', (req, res, next) => {  if (req.user) {  res.redirect('/');  return;  }  let login = req.body.login;  let fullname = req.body.fullname;  let password = req.body.password;  let password2 = req.body.password2;  let role = 0;  let hash = bcrypt.hashSync(req.body.password, 12);  if (req.user) {  res.redirect('/');  return;  }  if (req.body.password !== req.body.password2) {  res.redirect(`/auth/register?Error=Passwords+don't+match`);  } else {  let hash = bcrypt.hashSync(req.body.password, 12);  User.findByLogin(req.body.login, function (err, user) {  if (err) {  next(err);  } else if (user) {  res.redirect('/auth/register?Error=user+' + login + '+has+already+registered');  } else if (req.files.ava !== void 0) {  const user = new User(login, hash.toString(), fullname, role);  const buffer = req.files.ava[0].data;  // console.log("user.buffer:", buffer);  cloudUploader(buffer)  .then(photo => {  user.imageUrl = photo.secure\_url;  user.imageId = photo.public\_id;  console.log("User", user);  User.insert(user)  .then(user => {  const userId = (user.\_id).toString();  res.redirect('/auth/login');  })  .catch(err => res.status(500).send(err.toString()));  })  .catch(err => res.status(500).send(err.toString()));  }  else {  const user = new User(login, hash.toString(), fullname, role);  User.insert(user)  .then(user => {  const userId = (user.\_id).toString();  res.redirect('/auth/login');  })  .catch(err => res.status(500).send(err.toString()));  }  });  }  },  (req, res) => {  res.redirect('/auth/login');  }  ); |

Процес авторизації реалізован завдяки бібліотеці passport

|  |
| --- |
| router.post('/login',  passport.authenticate('local', {  failureRedirect: '/auth/login?Error=Passwors+or+login+incorrect,+please+try+again'  }),  (req, res) => {  res.redirect("/");  }  ); |

Щоби аутентифікувати користувача passport використовує стратегію

|  |
| --- |
| passport.use(new LocalStrategy({  usernameField: 'login',  passwordField: 'password'  }, function (login, password, done) {  User.findByLogin(login, (err, user) => {  if (err) {  return done(err);  }  if (!user) {  return done(null, false);  }  if (!bcrypt.compareSync(password, user.password)) {  return done(null, false);  }  return done(null, user);  });  })); |

1. Система розділяє права доступу до функціоналу сервісу: адміністратор, користувач, гість.
2. Система забезпечує адміністратора такими функціями :
   1. Управління профілями користувачів - адміністратор має можливість на сторінці із зареєстрованими користувачами редагувати та видаляти їх, а також змінювати роль обраного користувача у системі.

Перегляд користувача:

|  |
| --- |
| router.get('/:userId', check.checkAdmin, (req, res) => {  const userId = req.params.userId;  User.getById(userId)  .then(user => {  if (typeof user === 'undefined') {  res.status(404).send(`User with id ${userId} not found`);  } else {  res.render("user", {  "userRequired": user,  title: `User ${user.fullname}`,  'user': req.user  })  }  })  }); |

Зміна ролі звичайного користувача на користувача типу адміністратор

|  |
| --- |
| router.get('/:userId/setAdmin', check.checkAdmin, (req, res) => {  const userId = req.params.userId;  User.setAdmin(userId)  .then(() =>  res.redirect('/users')  )  .catch(err => res.status(500).send(err.toString()));  }); |

* 1. Редагування акордів та колекцій.

|  |
| --- |
| router.post('/:chordId/update', (req, res) => {  let chordId = req.params.chordId;  let name = req.body.name;  let tonality = req.body.tonality;  let notesQuantity = req.body.notesQuantity;  if (req.files.image !== void 0) {  const buffer = req.files.image[0].data;  cloud.cloudUploader(buffer)  .then(photo => {  let chord = Chord.getById(chordId)  .then(chord => {  chord.imageUrl = photo.secure\_url;  chord.imageId = photo.public\_id;  return Chord.update(chordId, name, tonality, notesQuantity, chord.imageUrl, chord.imageId);  })  })  .then(() => res.redirect(`/chords/${chordId}`))  .catch(err => {  console.log(err);  res.status(500).send(err.message)  });  }  }); |

* 1. Створення та видалення акордів та колекцій.

|  |
| --- |
| router.post('/:tabsId/delete', check.checkAuth, (req, res) => {  const tabsId = req.params.tabsId;  Tabs.delete(tabsId)  .then(tab => {  if (typeof tab === 'undefined') {  res.status(404).send(`Tab with id ${tabsId} not found`);  } else {  res.redirect(`/tabs`);  }  })  .catch(err => res.status(500).send(err.toString()));  }); |

|  |
| --- |
| router.post('/insert', check.checkAuth, (req, res) => {  const tab = new Tabs({  'name': req.body.name,  'autor': req.user.id,  });  const buffer = req.files.image.data;  cloudUploader(buffer)  .then(photo => {  tab.imageUrl = photo.secure\_url;  tab.imageId = photo.public\_id;  console.log(tab);  return Tabs.insert(tab);  })  .then(() => res.redirect('/tabs'))  .catch(err => res.status(500).send(err.message));  }  }); |

1. Система забезпечує користувача такими функціями :
   1. Перегляд, редагування та видалення власного профілю.
   2. Перегляд акордів та їх колекцій.
   3. Створення власних унікальних колекцій акордів.
2. Система виконує такі функції:
   1. Відображення списку всіх доданих акордів.

|  |
| --- |
| router.get('/', check.checkAuth, (req, res) => {  let page = parseInt(req.query.page);  if (!page) {  page = 1;  }  const byFilter = req.query.byFilter;  const chordsToDisplay = 4;  const startIndex = (page - 1) \* chordsToDisplay;  const endIndex = page \* chordsToDisplay;  Chord.getAll()  .then(chords => {  if (byFilter) {  chords = chords.filter((chord) =>  chord.name.toUpperCase().includes(byFilter.toUpperCase()));  }  chords = chords.slice(startIndex, endIndex);  res.render("chords", {  chords,  title: "All chords",  'user': req.user  })  })  .catch(err => res.status(500).send(err.toString()));  }); |

* 1. Відображення списку колекцій акордів.

|  |
| --- |
| router.get('/', check.checkAuth, (req, res) => {  let page = parseInt(req.query.page);  if (!page) {  page = 1;  }  const byFilter = req.query.byFilter;  const tabsToDisplay = 4;  const startIndex = (page - 1) \* tabsToDisplay;  const endIndex = page \* tabsToDisplay;  Tabs.getAll()  .find()  .populate("author")  .populate("chords.chord")  .exec()  .then(tabs => {  if (byFilter) {  tabs = tabs.filter((tab) =>  tab.name.toUpperCase().includes(byFilter.toUpperCase()));  }  tabs = tabs.slice(startIndex, endIndex);  res.render("tabs", {  tabs,  title: "All tabs",  'user': req.user  })  })  .catch(err => {  res.status(500).send(err.toString())  });  }); |

* 1. Відображення списку зареєстрованих у систему користувачів.
  2. Відображення обраного акорду з його властивостями (назва, тональність, кількість нот та зображення).
  3. Відображення обраної колекції та її властивостями (назва колекції, посилання на автора, зображення колекції, дата створення та масив акордів, що входять до даної колекції з посиланням на них).
  4. Відображення обраного користувача з повною інформацією про нього (ім’я, логін, дата реєстрації та фото).

1. Сайт містить форму зворотнього зв’язку, яку користувач може використовувати щоби зв’язатися із адміністратором сайту у разі виникнення будь-яких ускладнень при роботі з сайтом

**Організація даних**

Дані системи організовані всередині бази даних MongoDB у вигляді таблиць:

• **User** - авторизований користувач системи

• **login** - логін для подальшої авторизації на сайті

• **role** - роль користувача у системі (user or admin)

• **fullname** – повне ім’я користувача

• **password** – захешований пароль для авторизації на сайті

• **imageUrl** – посилання на аватар користувача у сервісі Cloudinary

• **registrationDate –** дата коли користувач зареєструвався

• **Chord -** основна сутність (акорд)

• **name –** назва акорду

• **tonality –** тональність

• **notesQuantity –** кількість нот

• **imageUrl –** посилання на зображення акорду у сервісі Cloudinary

• **uploadDate –** дата додання акорду

• **Tabs -** композиційна сутність (колекція акордів)

• **name –** назва колекції

• **autor –** id користувача, який створив колекцію

• **chords -** масив акордів, що входять до колекції

• **imageUrl –** посилання на зображення колекції у сервісі Cloudinary

• **created –** дата створення колекції

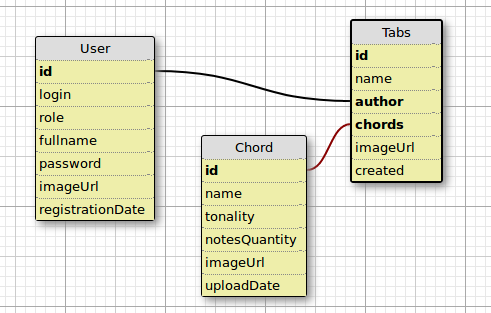
****

Рис. Таблиці та зв’язки бази даних

**Приклади даних:**

****

Рис. User

****

Рис. Tabs

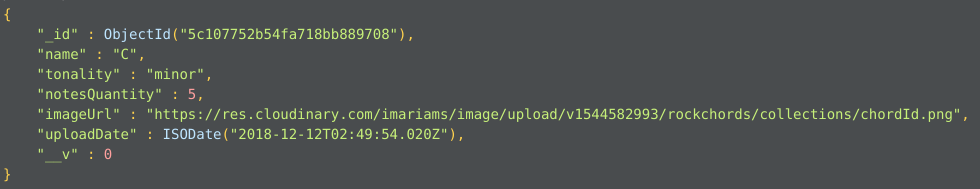
****

Рис. Chord

**Структура web-сайту**

Структура сайту передбачає такі рівні:

Головна сторінка сайту із доступними для користувача сторінками

Сторінка Users (доступна лише користувачу типу admin) - сторінка для перегляду списку всіх користувачів з можливістю переходу до конкретного обраного користувача

Сторінка Chords - сторінка для перегляду списку всіх акордів та переходу до конкретного акорду

Сторінка Tabs - сторінка для перегляду списку всіх колекцій та переходу до конкретної колекції акордів

Сторінка Profile - сторінка персонального профілю користувача

Сторінка Chord - сторінка з детальною інформацією про обраний акорд

Сторінка Tab - сторінка з детальною інформацією про обрану колекцію акордів

Сторінка User - сторінка з детальною інформацією про певного користувача

Сторінка New chord - сторінка для створення нового акорду

Сторінка New tab - сторінка для створення нової колекції акордів

Сторінка About - сторінка для перегляду додаткової інформації про сайт

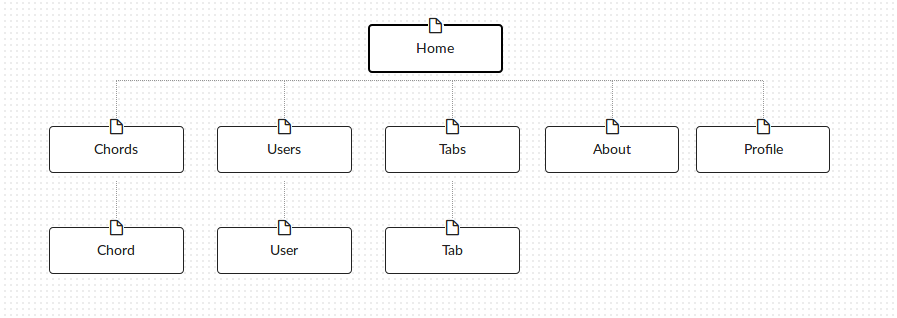


Рис. Карта сайту

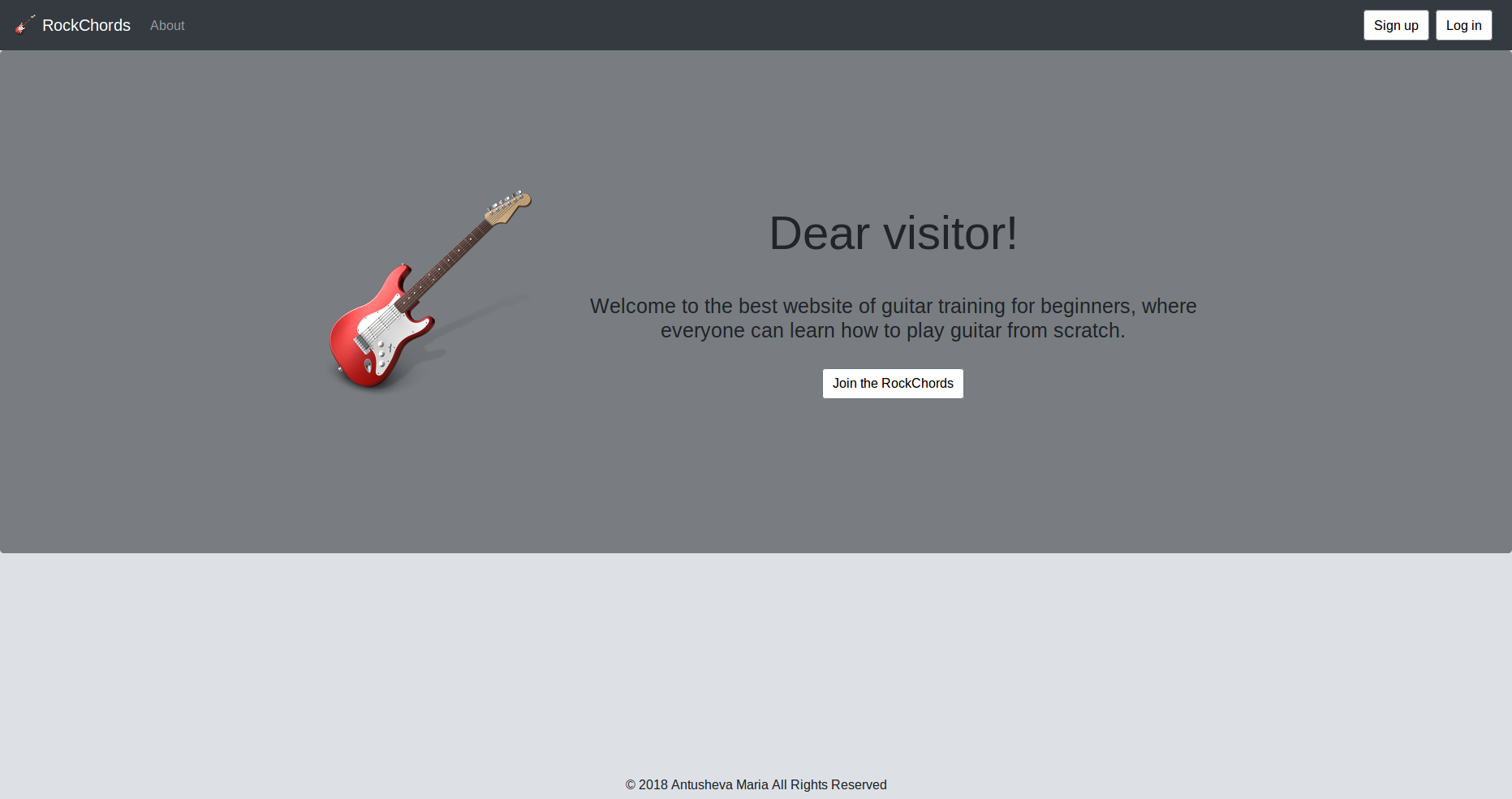
**Вигляд сторінок web-сайту**

Рис. Головна сторінка

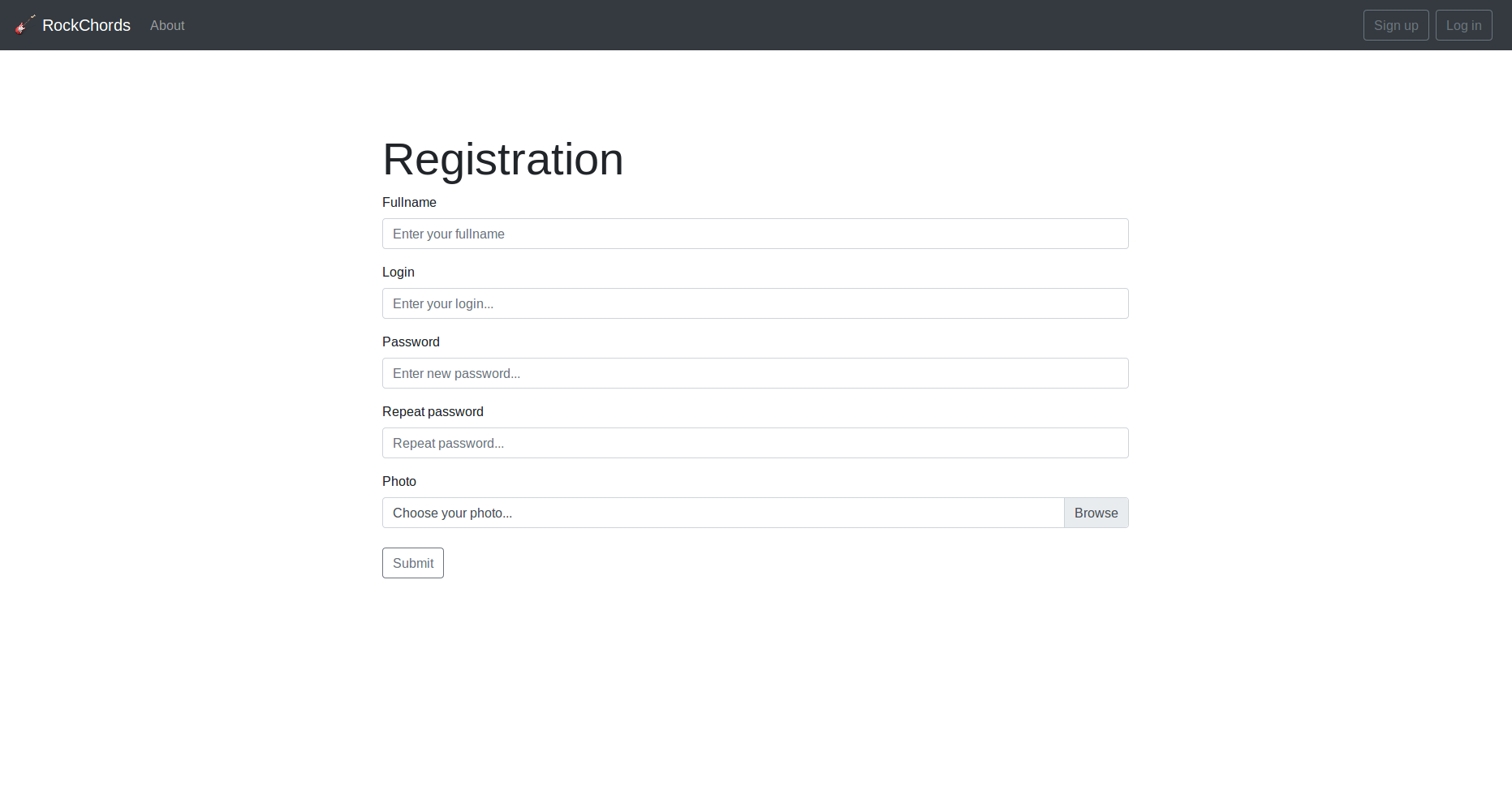


Рис. Сторінка реєстрації

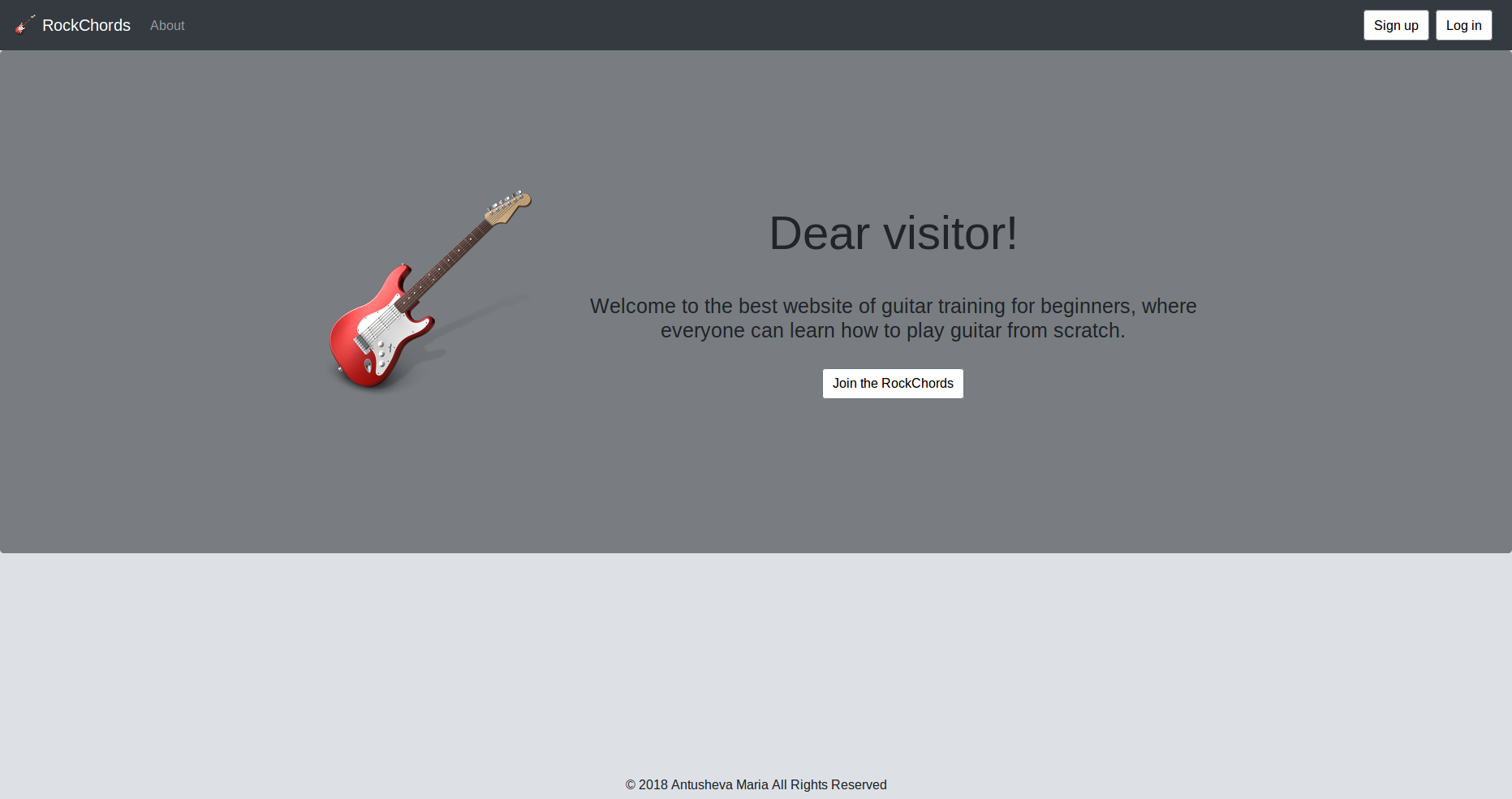


Рис. Сторінка авторизації

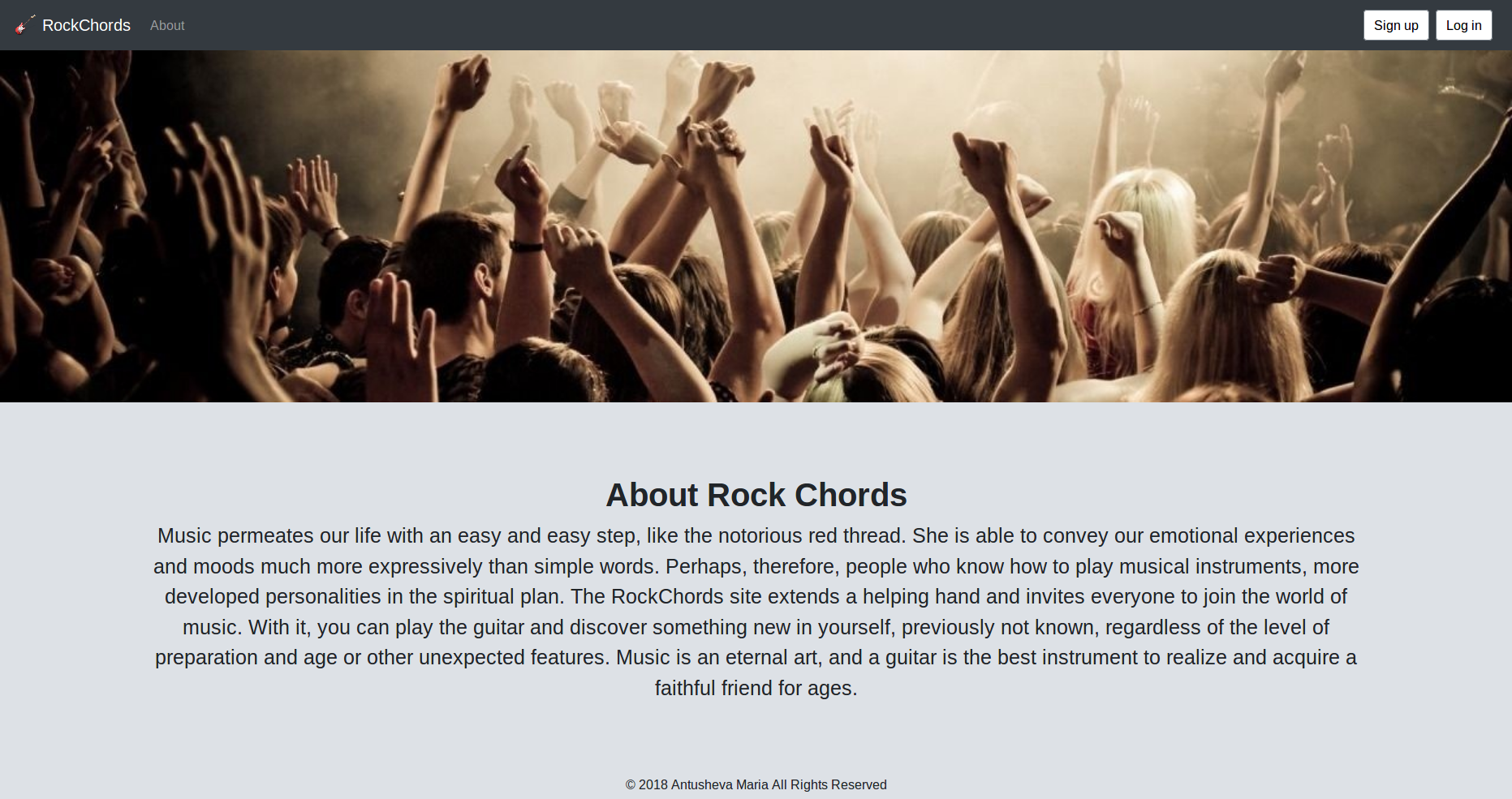


Рис. Сторінка додаткової інформації про сайт

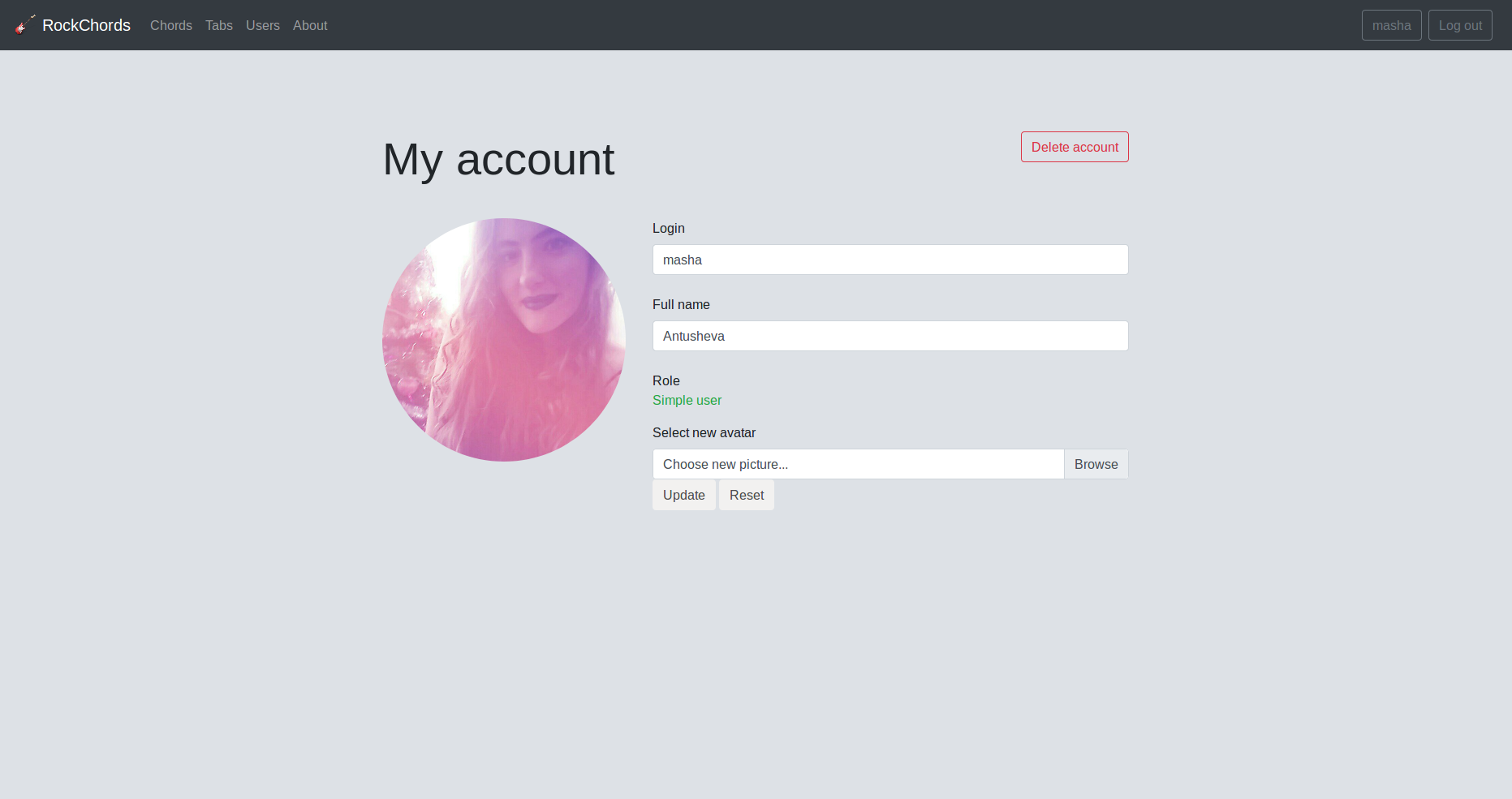
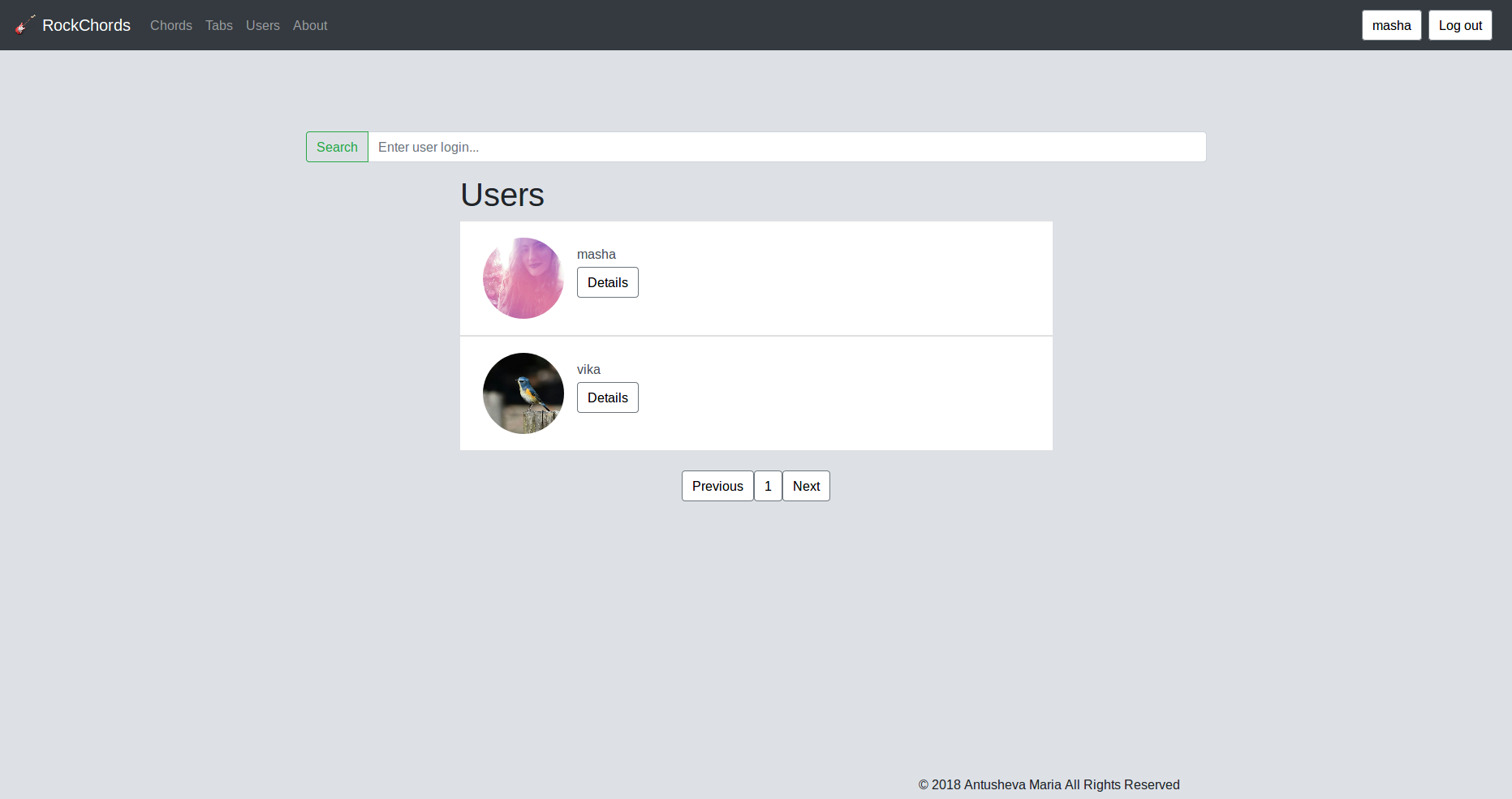


Рис. Профіль користувача



Сторінка з усіма користувачами(доступна лише користувачу типу адмін)

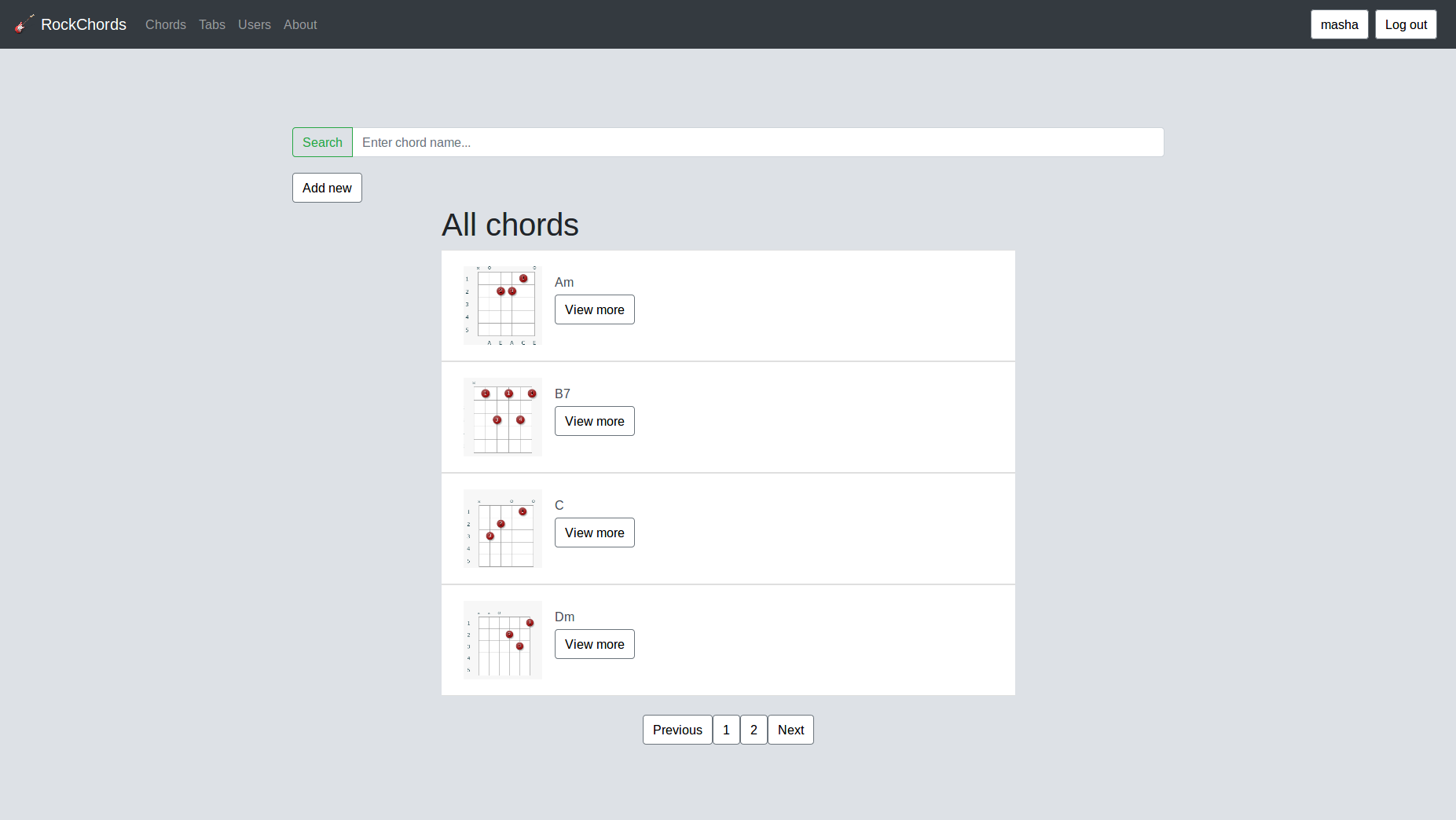
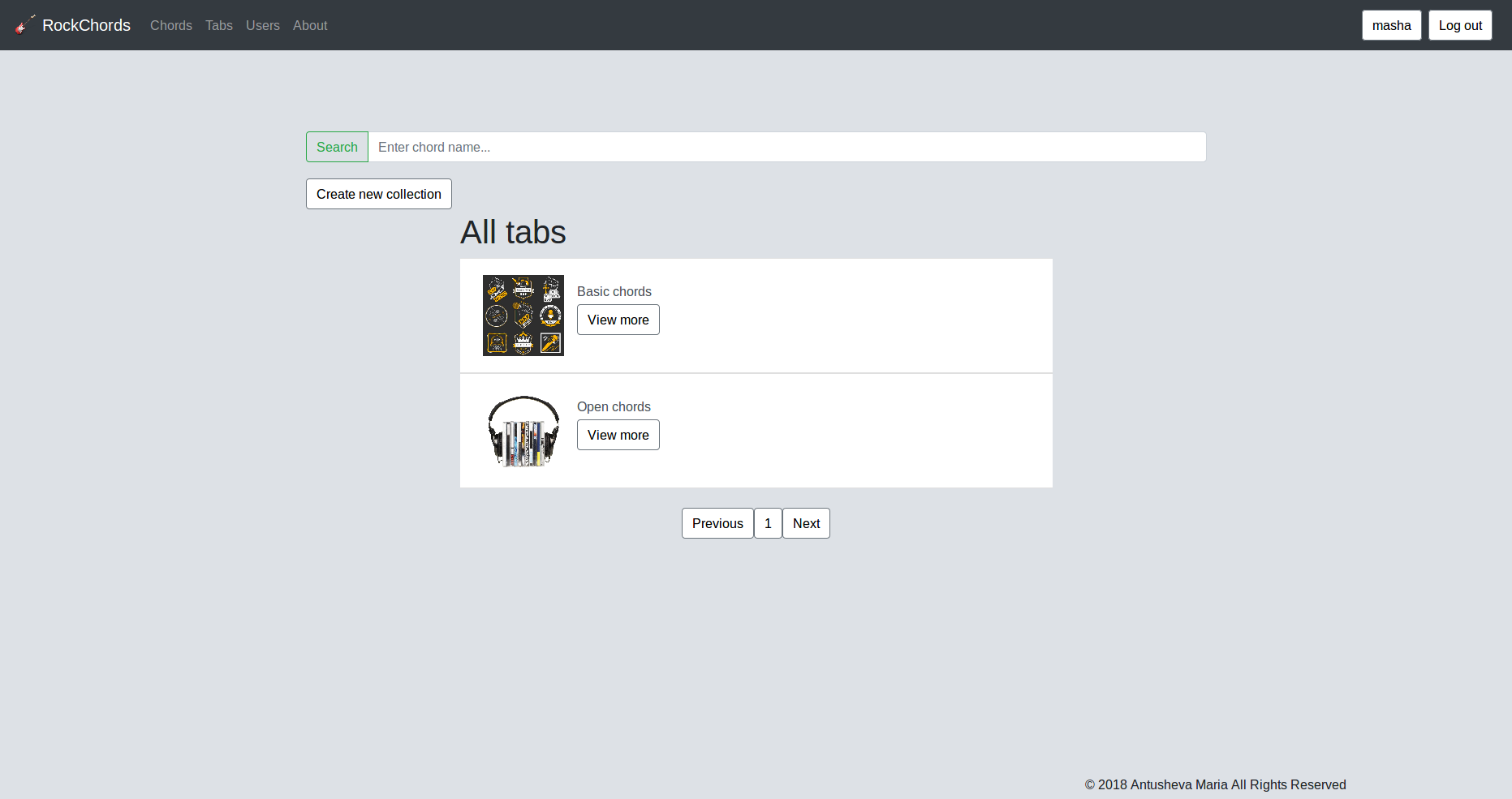


Рис. Сторінка з усіма акордами



Сторінка з колекціями акордів

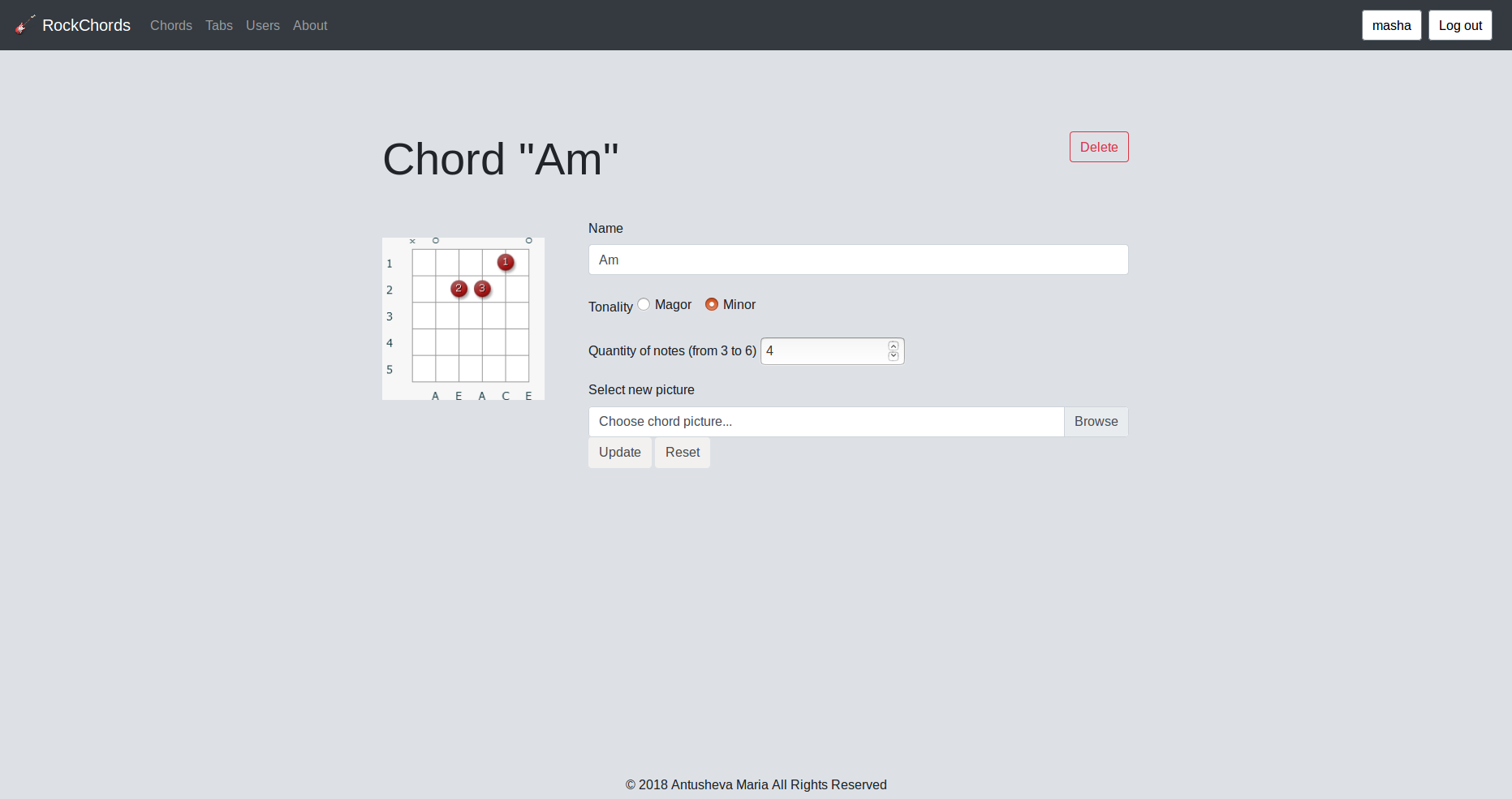


Рис. Сторінка акорду

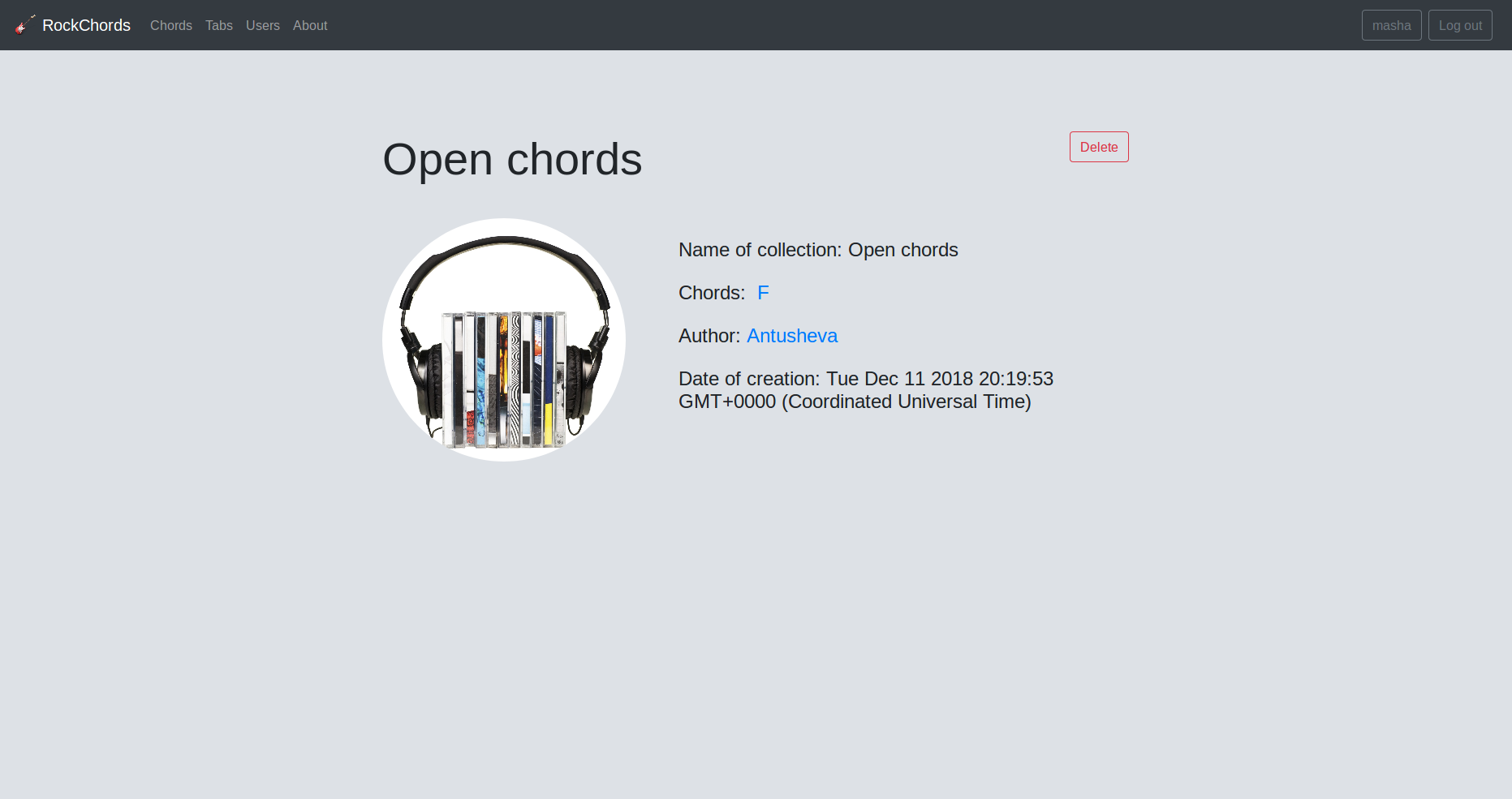


Рис. Сторінка колекції

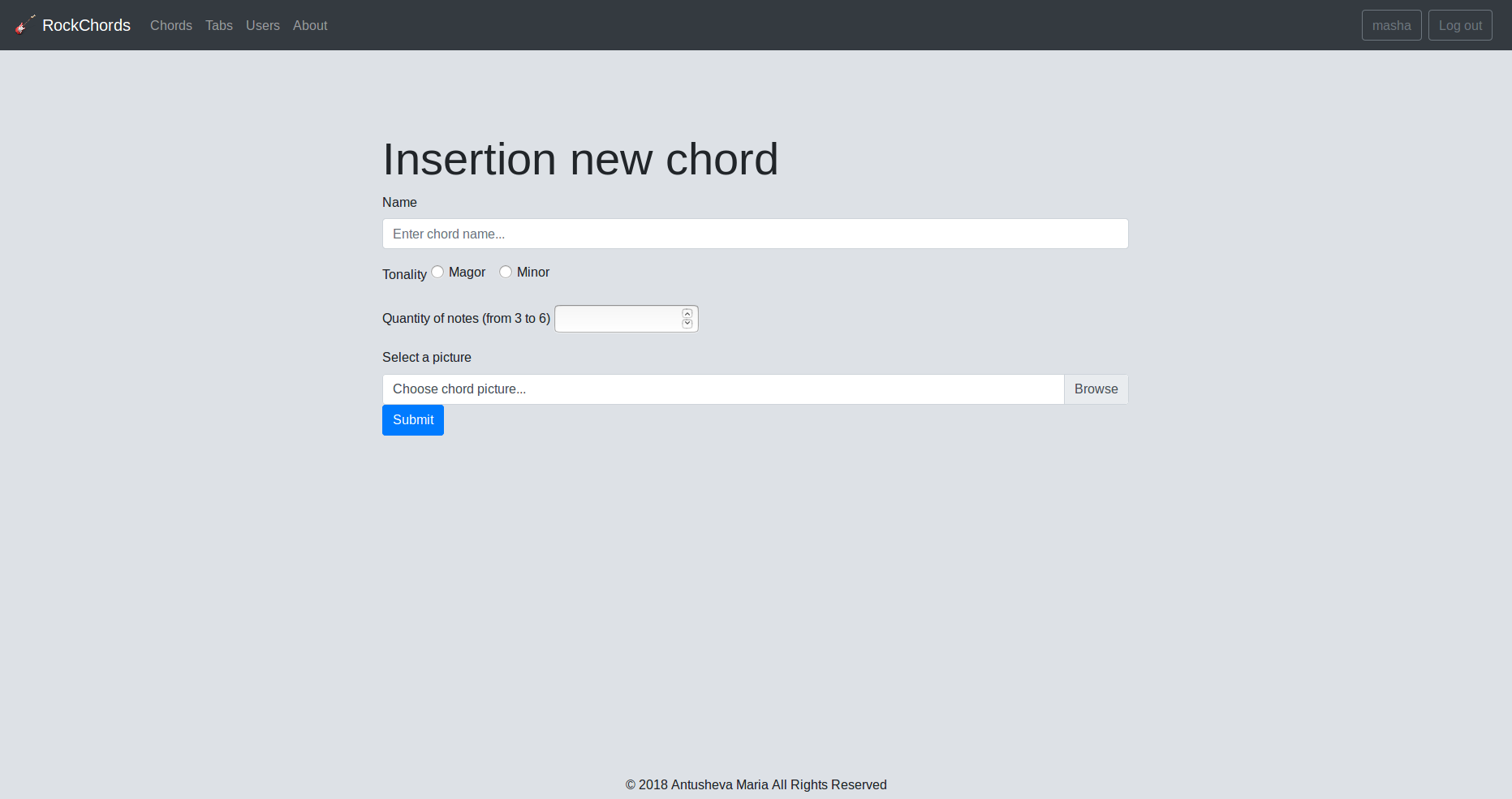


Рис. Сторінка створення нового акорду

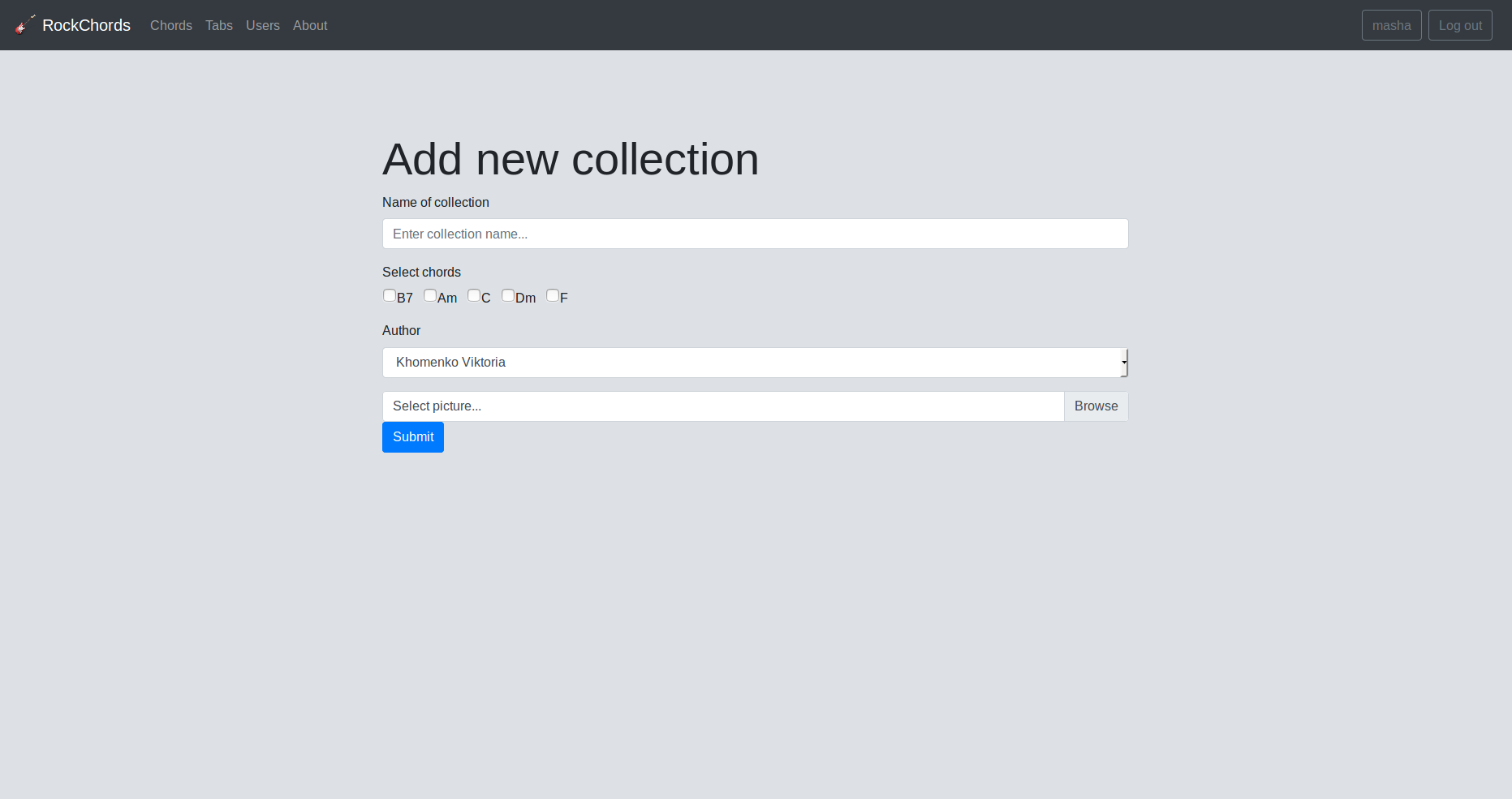


Рис. Сторінка створення нової колекції

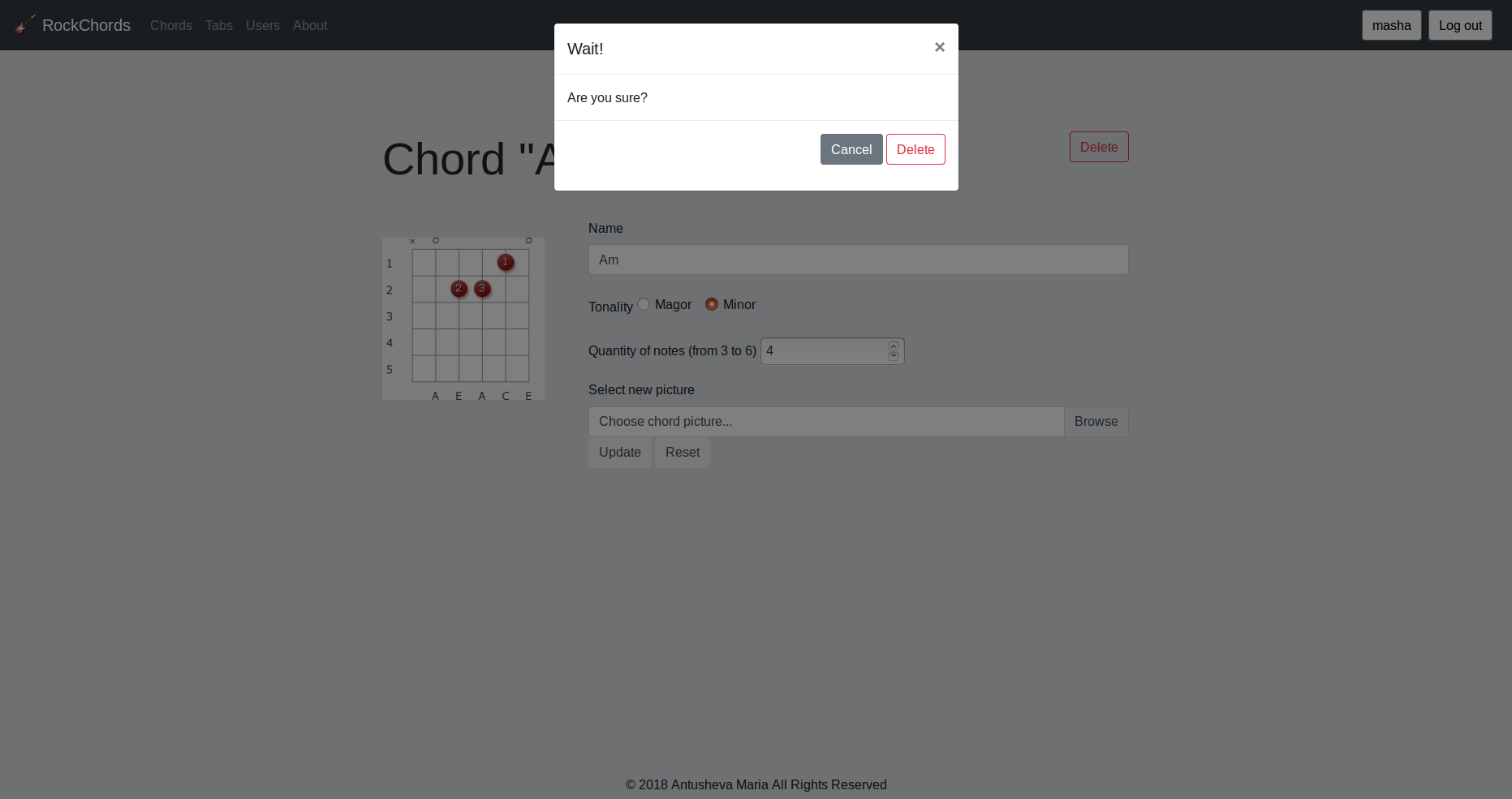


Рис. Підтвердження видалення акорду

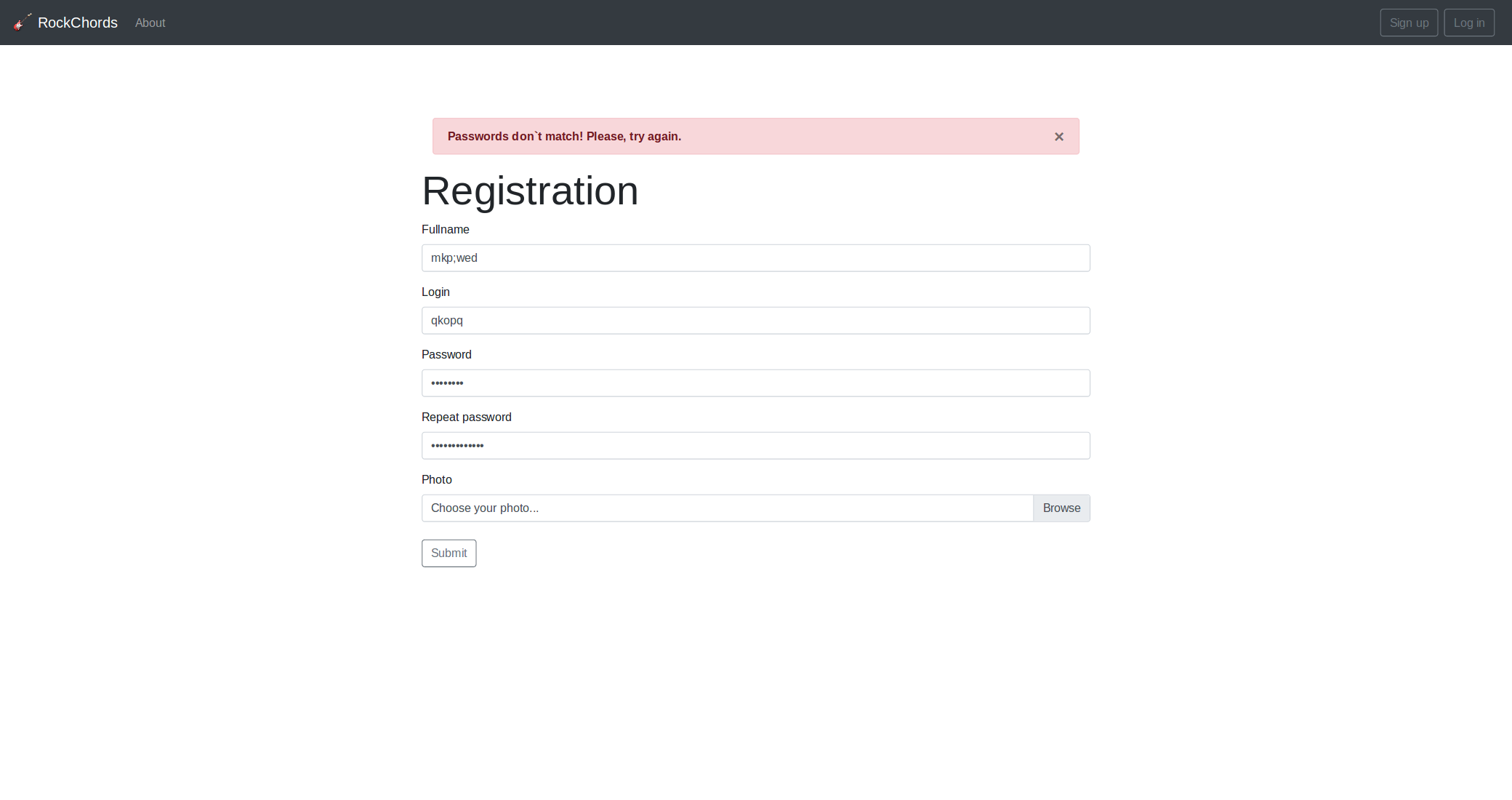


Рис. Некоректні дані для реєстрації

**Висновки**

Під час виконання даної курсової роботи, я ознайомилася з поняттям веб-документів та технологіями їх створення та стилізації. Використала мову програмування JavaScript та середовище NodeJS. Протягом роботи я навчилася створювати модулі, синхронно працювати з файловими потоками та обробляти JSON-строки. Для генерації контенту ознайомилася із шаблонізаторами веб-сторінок, особливо із шаблонізатором Mustache, і створила простий веб-сервер, що надає доступ до статичних документів. Згодом щоби реалізувати взаємодію користувача з веб-сайтом навчилася створювати HTML-форми та обробляти на сервері внесені користувачем даніб а також вивчила основні принципи асинхронного програмування за допомогою callback-функцій. Вмію працювати із базами даних у веб-сервері та використовувати Promise-об'єкти при написанні асинхронного коду. Підготувала свій власний веб-проект для розгортання на віддаленому серверному хості та відкрила для нього доступ до мережі Інтернет, працюючи при цьому з віддаленою базою даних, і, згодом, познайомилася із хостингом Cloudinary для завантаження відео та зображень для подальшого використання на сайті. Реалізувала персоналізацію вмісту сайту за допомогою користувачів і авторизацію з виконанням сесій. Створила JSON REST API для доступу до ресурсів веб-сервера. Навчилася створювати динамічні веб-сторінки за допомогою браузерних скриптових мов.

Надалі я планую розвивати свій сайт : для кращої взаємодії користувача необхідно додати дизайнерські рішення з приводу оформлення сторінок; додати можливість оцінювати колекції лайками та сортувати іх за стилем, автором; створити нову сутність - Композицію, яку можна буде прослуховувати, переглядати ії текст та акорди, що в ній використовуються.

Отже, за допомогою мови програмування JavaScript та певних фреймворків, я створила цікавий сайт, який планую надалі розвивати.