**VISOKA TEHNIČKA ŠKOLA**

**STRUKOVNIH STUDIJA**

**S** **U B O T I C A**

**PROJEKAT**

**iz predmeta Web programiranje**

**KANDIDAT MENTOR**

**Bojan Knežević dr Zlatko Čović**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SUBOTICA 2023.god.**

**SADRŽAJ**

[OPIS ZADATKA 3](#_Toc20668)

[REALIZACIJA ZADATKA 4](#_Toc11184)

[STRUKTURA BAZE PODATAKA 15](#_Toc13639)

[OPIS FUNKCIONALNOSTI 16](#_Toc26460)

[KORIŠĆENA LITERATURA 24](#_Toc18942)

# OPIS ZADATKA

Cilj projekta mog tima je bio da kreiramo web aplikacija, odnosno sajt za rezervaciju stolova u jednom restoranu.

Aplikacija razlikuje četiri nivoa pristupa: gost, registrovan/prijavljen korisnik, radnik restorana i administrator. Svaki od nivoa ima određene mogućnosti i ograničenja koja su nam zadata u opisu projekta. Gost može vidi osnovni sadržaj aplikacije i kreira nalog, prijavljen korisnik može da pravi i otkazuje rezervacije, radnik može da menja status rezervacije odnosno da li su gosti stigli, naručili i druge. I naravno administrator koji može da vidi sve korisničke i naloge radnika. Takođe može da menja informacije o stolovima i vidi sve rezervacije koje se nalaze u bazi podataka.

# REALIZACIJA ZADATKA

Realizacija projekta je bila veoma interesantna i poučna za mene. Imao sam priliku da iskušam svoje znanje i sposobnosti kako sa predavanja tako sa BizKod-a na kom sam učestvovao.

Tehnologije koje sam koristio su: HTML, CSS, JavaScript, jQuery, Fetch API, JSON, Bootstrap, PHP i MySQL.

Delovi nekih od PHP i Javascript kodova i kratak opis o tome šta radi:

Signup.php:

Opis: Uzima se input iz forme, šalje se email sa verifikacionim linkom unutar koga je i verifikacioni kod koji biva generisan. Takođe input se čisti kako bi se sprečili XSS napadi.

Kod:   
<?php

require "template/header.php";

require "db\_config.php";

require "functions.php";

If(isset($\_SESSION['logged\_in'])&& $\_SESSION['logged\_in']) {

header("location: index");

exit();

}

$conn = connect();

$timeout = 60; //1 minute

use PHPMailer\PHPMailer\PHPMailer;

require\_once "PHPMailer-6.8.0/vendor/autoload.php";

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

//Filtered inputs

$fname = filter\_input(INPUT\_POST, 'fname', FILTER\_UNSAFE\_RAW);

$lname = filter\_input(INPUT\_POST, 'lname', FILTER\_UNSAFE\_RAW);

$email = filter\_input(INPUT\_POST, 'email', FILTER\_VALIDATE\_EMAIL);

$gender = $\_POST['gender'];

$rawDate = $\_POST['datepicker'];

$phone = filter\_input(INPUT\_POST, 'phone', FILTER\_UNSAFE\_RAW);

$errorMessage = '';

$date = formatDate($rawDate);

// Check if the email is in timeout (wip)

$lastRequest = isset($\_SESSION['last\_request']) ? $\_SESSION['last\_request'] : 0;

if (time() - $lastRequest < $timeout) {

$errorMessage = "Please wait before making another request.";

}

else {

$errorMessage ='';

if (!validEmail($email)) {

$errorMessage = "There is already an account registered under that email!";

} elseif (!validDOB($date)) {

$errorMessage = "You must be at least 13 years old to make an account!";

} else {

$stmt = $conn->prepare("INSERT INTO email\_requests (email, last\_request) VALUES (?, NOW()) ON DUPLICATE KEY UPDATE last\_request = NOW()");

$stmt->execute([$email]);

$\_SESSION['fname'] = $fname;

$\_SESSION['lname'] = $lname;

$\_SESSION['email'] = $email;

$\_SESSION['gender'] = $gender;

$\_SESSION['date'] = $date;

$\_SESSION['phone'] = $phone;

$verificationCode = generateVerificationCode();

$\_SESSION['verificationCode'] = $verificationCode;

//UPDATE EMAIL FOR SERVER! on reservation.php as well

// PHPMailer settings

$mail = new PHPMailer();

$mail->isSMTP();

$mail->Host = 'sandbox.smtp.mailtrap.io';

$mail->SMTPAuth = true;

$mail->Port = 2525;

$mail->Username = '144ca3a32ec68b';

$mail->Password = '19c9a32187fb89';

// Sender and recipient settings

$mail->setFrom('noreply@milfirefox.com', 'Man I Love Food');

$mail->addAddress($email, $fname . ' ' . $lname);

// Email subject and body with verification link

$mail->Subject = 'Account creation';

$verificationLink = generateVerificationLink($email, $verificationCode);

$mail->Body = "Hi $fname $lname,\n\nThank you for signing up! Please click on the following link to confirm your email and set your password:\n\n$verificationLink\n\nSincerely,\nMan I Love Food Website Team";

if ($mail->send()) {

$\_SESSION['signup'] = true; // Set the signup flag

$errorMessage = "Confirmation email sent successfully!";

} else {

echo 'Mailer Error: ' . $mail->ErrorInfo;

}

}

$\_SESSION['last\_request'] = time();

}

}

?>

verification.php:

Opis: Putem varijabli secije, u nastavnu “session vaijabli”, informacije se šalju od signup.php do verification.php gde se takođe traži input za lozinku i potvrda upisane lozinke. Takođe complexity\_check.js je ovde implementiran, više o njemu nakon kod

Kod:

$\_SESSION['verified'] = false;

$fname = $\_SESSION['fname'];

$lname = $\_SESSION['lname'];

$email = $\_SESSION['email'];

$gender = $\_SESSION['gender'];

$date = $\_SESSION['date'];

$phone = $\_SESSION['phone'];

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$password = $\_POST['password'];

$confirmPassword = $\_POST['confirm\_password'];

if ($password !== $confirmPassword) {

$errorMessage = "Password and confirm password do not match!";

} else {

try {

$conn = connect();

$hashedPassword = password\_hash($password, PASSWORD\_BCRYPT);

// SQL statment

$stmt = $conn->prepare("INSERT INTO accounts (first\_name, last\_name, email, gender, date\_of\_birth, phone, password, created\_at)

VALUES (:fname, :lname, :email, :gender, :date\_of\_birth, :phone, :password, :created\_at)");

$stmt->bindParam(':fname', $fname);

$stmt->bindParam(':lname', $lname);

$stmt->bindParam(':email', $email);

$stmt->bindParam(':gender', $gender);

$stmt->bindParam(':date\_of\_birth', $date);

$stmt->bindParam(':phone', $phone);

$stmt->bindParam(':password', $hashedPassword);

$stmt->bindValue(':created\_at', date('Y-m-d H:i:s'));

$stmt->execute();

$statement = $conn->prepare("SELECT id FROM accounts WHERE email=?");

$statement->execute([$email]);

$resultAccountId = $statement->fetch(PDO::FETCH\_ASSOC);

$id = $resultAccountId['id'];

$stmt = $conn->prepare("INSERT INTO user\_role SET account\_id=?, role\_id=3");

if ($stmt->execute([$id])) {

unset($\_SESSION['signup']);

unset($\_SESSION['fname']);

unset($\_SESSION['lname']);

unset($\_SESSION['email']);

unset($\_SESSION['gender']);

unset($\_SESSION['date']);

unset($\_SESSION['phone']);

$\_SESSION['verified'] = true;

$errorMessage = "Account successfully verified!";

} else {

$\_SESSION['verified'] = false;

}

} catch (PDOException $e) {

$errorMessage = "Error: " . $e->getMessage();

}

}

Complexity\_check.js:  
Opis: Ovde se nalaze 2 funkcije koje su implementirane u verification.php kod unosa šifre, to jest na pormenu unešene šifre. Prva funkcija je samo za ažuriranje boje, a druga funkcija prima json podatke iz password\_strength.php gde se izvršava provera šifre.

Kod:

function updatePasswordInputColor(isStrong) {

const passwordInput = document.getElementById("password");

if (isStrong) {

passwordInput.classList.remove("password-weak");

passwordInput.classList.add("password-strong");

} else {

passwordInput.classList.remove("password-strong");

passwordInput.classList.add("password-weak");

}

}

function checkPasswordStrength(passwordInput) {

let password = passwordInput.value;

fetch("password\_strength.php", {

method: "POST",

headers: {

"Content-Type": "application/json"

},

body: JSON.stringify({ password: password })

})

.then(response => response.json())

.then(data => {

updatePasswordInputColor(data.complexity === "Strong");

})

.catch(error => {

console.error("Error checking password strength:", error);

updatePasswordInputColor(false);

});

}

admin\_dash.js:

Opis: U ovom delu koda se realizuje dodavanje novih radnika i korisnika od strane administratora. Na osnovu unosa podataka od strane administratora, u nastavku admin, u formi iz admin.php i uz pomoć JSON šalje se na add\_worker.php gde se novi radnik, odnosno korisnik upisuje u bazu podataka. Mogućnost da admin dodaje nove korisnike sam prvobitno ubacio čisto zbog testiranja ali sam odlučio da ostavim u verziji koju predajem i koja je na serveru kao dodatak pošto se to nije tražilo u zadatku.

Kod:

//Add worker

const addWorkerButton = document.querySelector(".btnAddWorker");

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

addWorkerButton.addEventListener("click", () => {

addWorkerForm.style.display = "block"; // Show the form

addWorkerForm.reset();

addWorkerButton.style.display = "none";

});

addWorkerForm.addEventListener("submit", async (event) => {

event.preventDefault();

const data = {};

const inputs = addWorkerForm.querySelectorAll("input[name='workerId'], input[name='first\_name'], input[name='last\_name'], input[name='email'], select[name='gender'], input[name='date\_of\_birth'], input[name='phone'], input[name='password']");

inputs.forEach((input) => {

const key = input.getAttribute("name");

data[key] = input.value;

});

try {

console.log(data);

const response = await fetch("add\_worker.php", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify(data),

});

if (response.ok) {

addWorkerButton.style.display = "block";

addWorkerForm.reset();

addWorkerForm.style.display = "none"; // Hide the form again

}

} catch (error) {

console.log(error)

}

});

functions.php:

Kako samo ime kaže, u ovom delu koda se nalaze funkcije.

<?php

//Generate a random verification code

function generateVerificationCode($length = 20): string

{

$characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';

$code = '';

for ($i = 0; $i < $length; $i++) {

$code .= $characters[rand(0, strlen($characters) - 1)];

}

return $code;

}

//Date formatting

function formatDate($inputDate): false|string

{

$dateObj = DateTime::createFromFormat('d-m-Y', $inputDate);

if($dateObj){

return $dateObj->format('Y-m-d');

}

else{

return false;

}

}

//Same as last just reverse

function reverseFormatDate($inputDate): false|string

{

$dateObj = DateTime::createFromFormat('Y-m-d', $inputDate);

if($dateObj) {

return $dateObj->format('d-m-Y');

}

else{

return false;

}

}

//Check if email is already used for a different account

function validEmail($email): bool

{

require\_once "db\_config.php";

$conn = connect();

$stmt = $conn->prepare("SELECT email FROM accounts WHERE email=?");

$stmt -> execute([$email]);

$result = $stmt->fetch();

if($result != NULL){

return false; //is NOT valid

}

return true; // is valid

}

//Date of birth validation valid if(13>=DOB<=130)

function validDOB($dob): bool

{

$dob = new DateTime($dob);

$min = new DateTime();

$min->sub(new DateInterval('P13Y')); // Subtract 13 years

$max = new DateTime();

$max->sub(new DateInterval('P130Y')); // Subtract 130 years

$diffToMin = $dob->diff($min);

$diffToMax = $dob->diff($max);

return $diffToMin->invert === 0 && $diffToMax->invert === 1;

}

// Check if the date is not in the past or today

function validDate($date): bool

{

$selectedDate = new DateTime($date);

$today = new DateTime();

if($selectedDate > $today){

return true;

}

return false;

}

function getReservationTime($time): false|string

{

return match ($time) {

'Breakfast' => '07:00',

'Lunch' => '12:00',

'Dinner' => '18:00',

default => false,

};

}

function getSeats($guests): int

{

if($guests < 3){$seatsNeeded = 2;}

elseif ($guests < 5){$seatsNeeded = 4;}

else($seatsNeeded = 8);

return $seatsNeeded;

}

//Retrieves user role from database

function getUserRoleFromDB($email, $password): ?string

{

$conn = connect();

$stmt = $conn->prepare("SELECT id, password FROM accounts WHERE email = ?");

$stmt->execute([$email]);

$result = $stmt->fetch();

if ($result && password\_verify($password, $result['password'])) {

$stmt = $conn->prepare("SELECT role\_id FROM user\_role WHERE account\_id = ?");

$stmt->execute([$result['id']]);

$role = $stmt->fetch();

$stmt = $conn->prepare("SELECT id FROM accounts WHERE email = ?");

$stmt->execute([$email]);

$id = $stmt->fetch();

$\_SESSION['id'] = $id;

switch ($role['role\_id']) {

case 1:

return "admin";

case 2:

return "worker";

default:

return "user";

}

} else {

return null;

}

}

//Function to generate a verification link

function generateVerificationLink($email, $verificationCode): string

{

return "https://milfirefox.stud.vts.su.ac.rs/verification.php?email=" . urlencode($email) . "&code=" . urlencode($verificationCode);

}

function getIPAddress()

{

//ip is from the share internet

if (!empty($\_SERVER['HTTP\_CLIENT\_IP'])) {

$ip = $\_SERVER['HTTP\_CLIENT\_IP'];

}//ip is from the proxy

elseif (!empty($\_SERVER['HTTP\_X\_FORWARDED\_FOR'])) {

$ip = $\_SERVER['HTTP\_X\_FORWARDED\_FOR'];

}//ip is from the remote address

else {

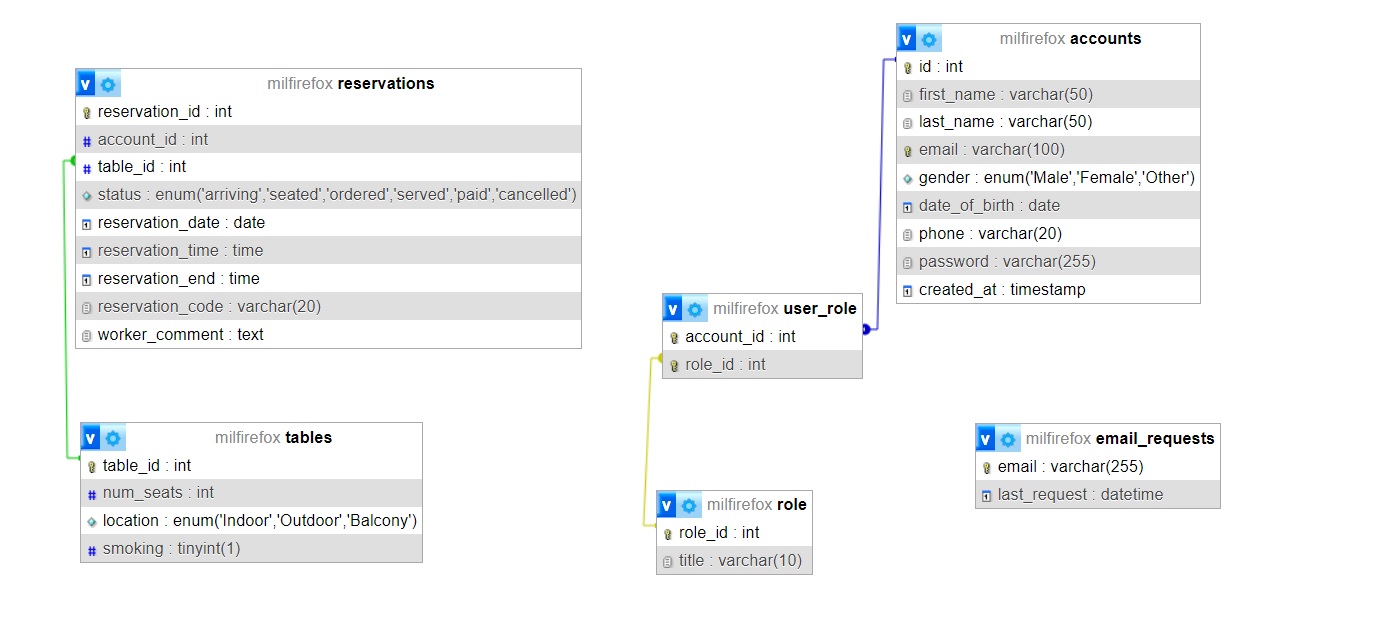
$ip = $\_SERVER['REMOTE\_ADDR'];

}

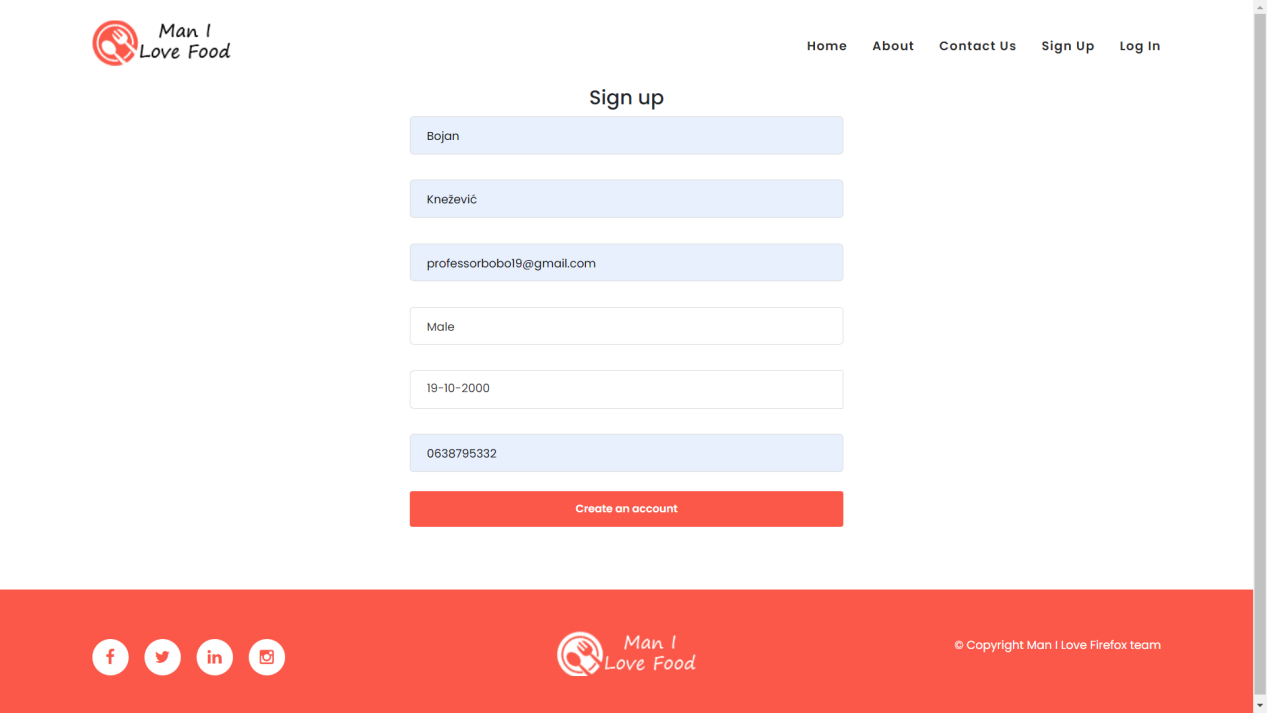
return $ip;

}

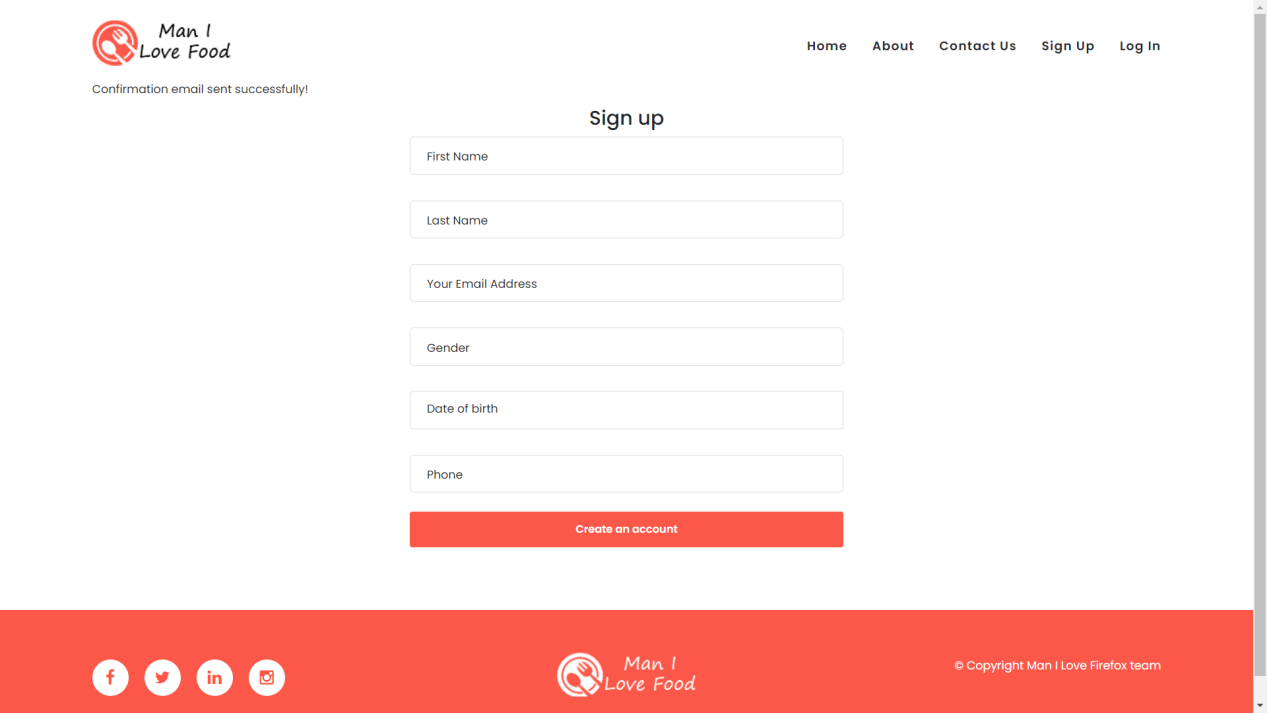
# STRUKTURA BAZE PODATAKA

****

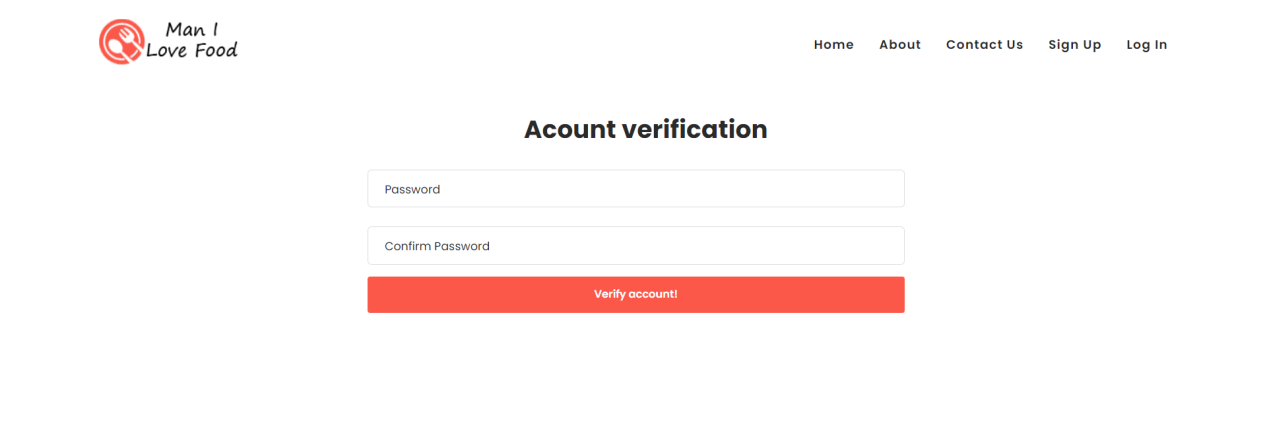
# OPIS FUNKCIONALNOSTI

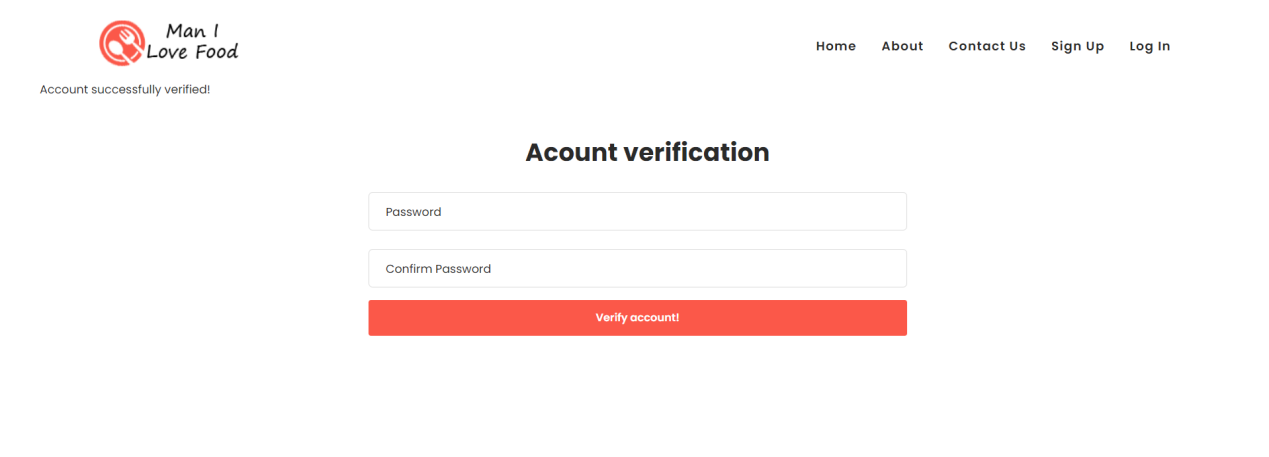
Prijava novog korisnika. Ovde se upisuju detalji prilikom kreiranje novog korisnika. 

Ovako izgleda kad je prijava uspešna.

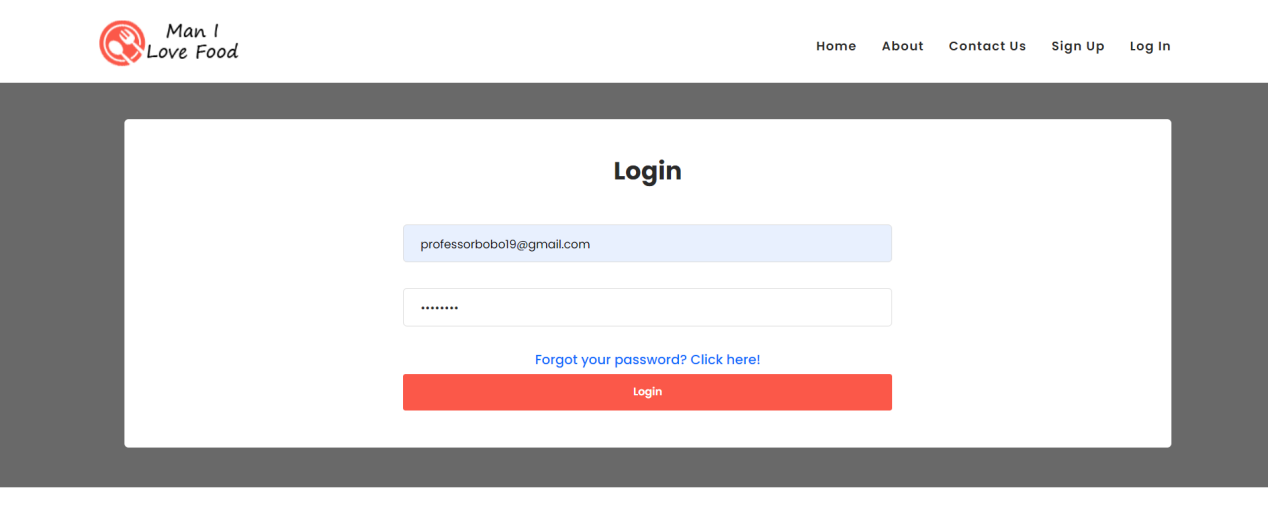


Nakon klika na link koji je poslat putem email-a bivamo preusmereni na ***verification*** odnosno krajni korak pravljenja novog profila.

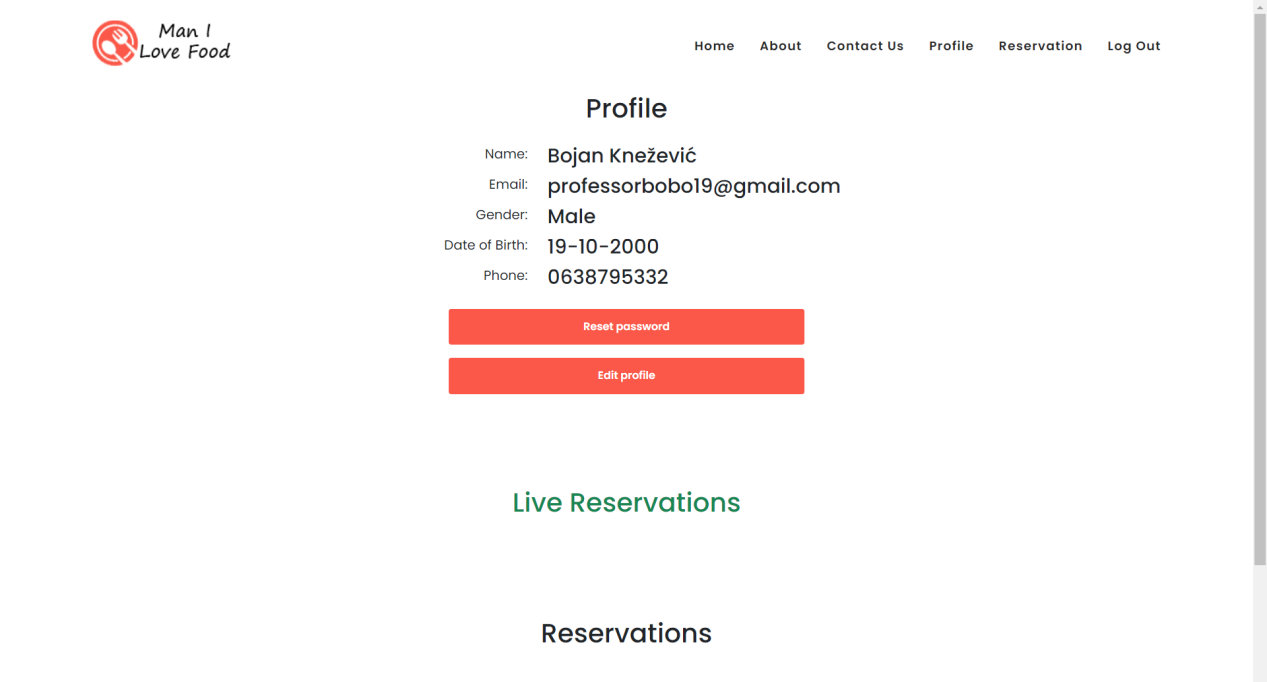




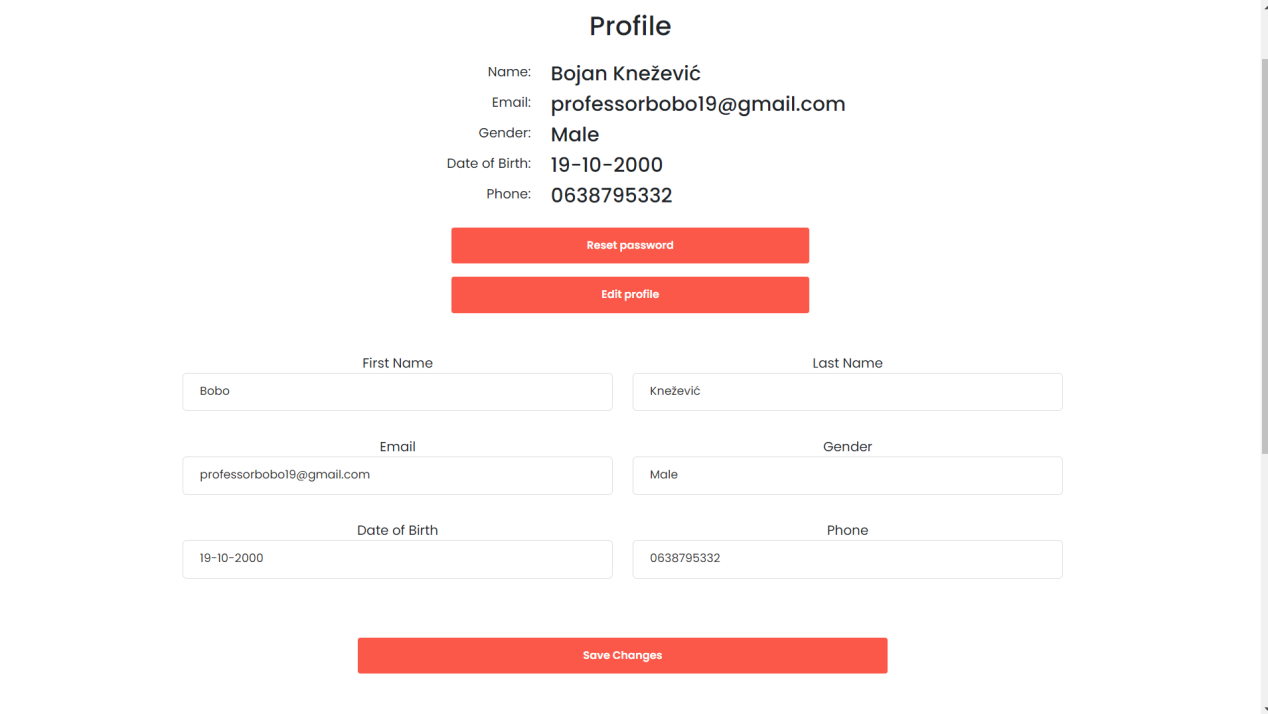
Prijava registrovanog korisnika klikom na ***Log in.***



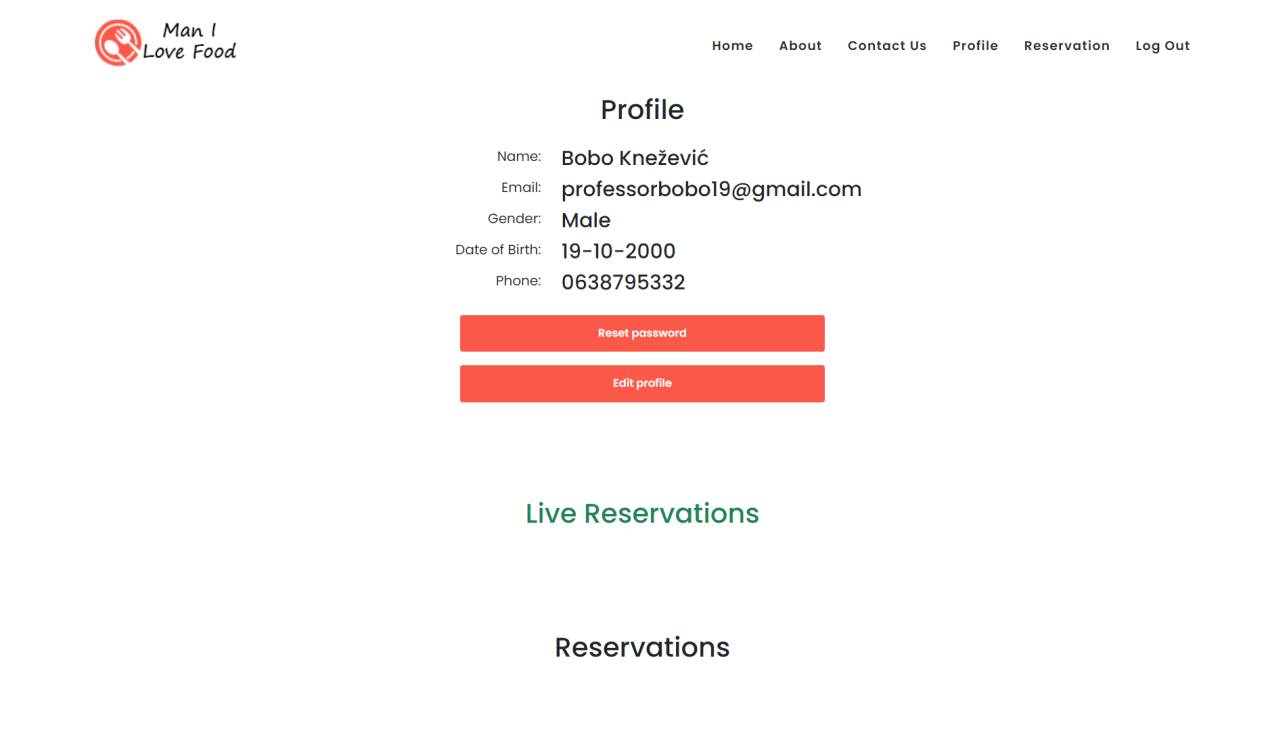
Profil korisnika gde može da vidi svoje informacije, da ih menja, i vidi rezervacije koje su u budućnosti koje može da otkaže i trenutne koje se ne mogu otkazati.



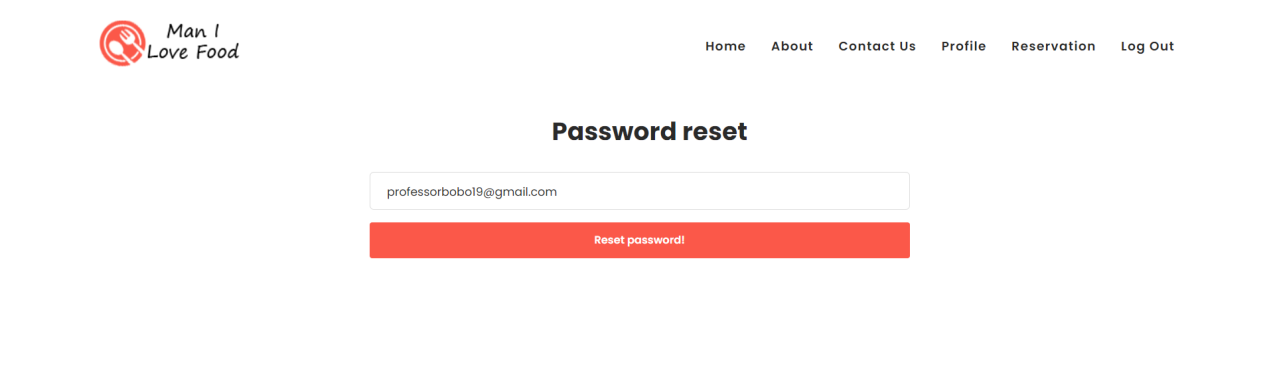
Izmena detalja profila(promena imena iz Bojan u Bobo)



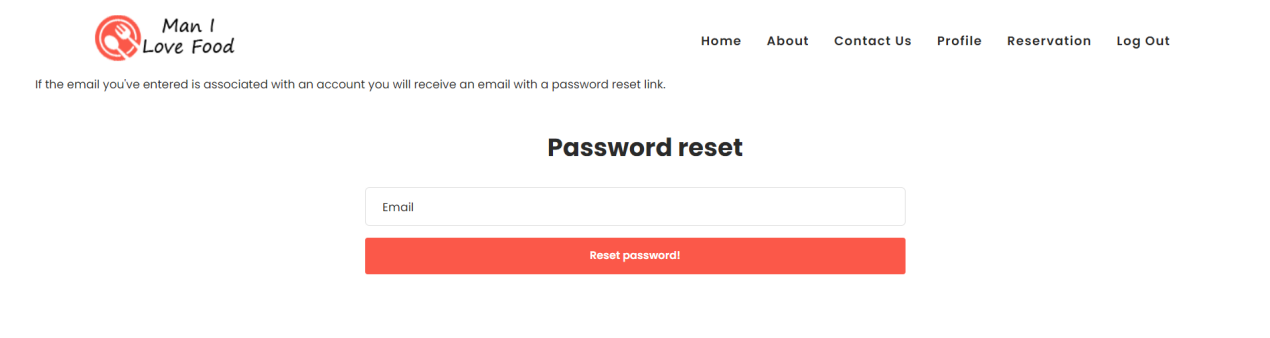
Izmena detalja profila(nakon promene novi podaci bivaju unešeni u bazu i se osvežava stranica)



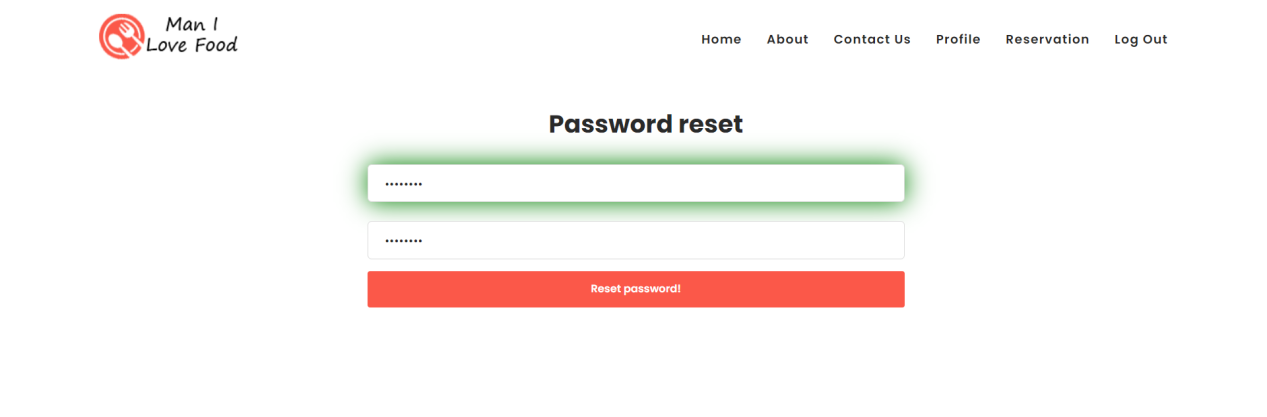
Promena lozinke nakon klika na **Reset password** na profilu ili na ***Forgot your password? Click here!*** na **Log in** stranici



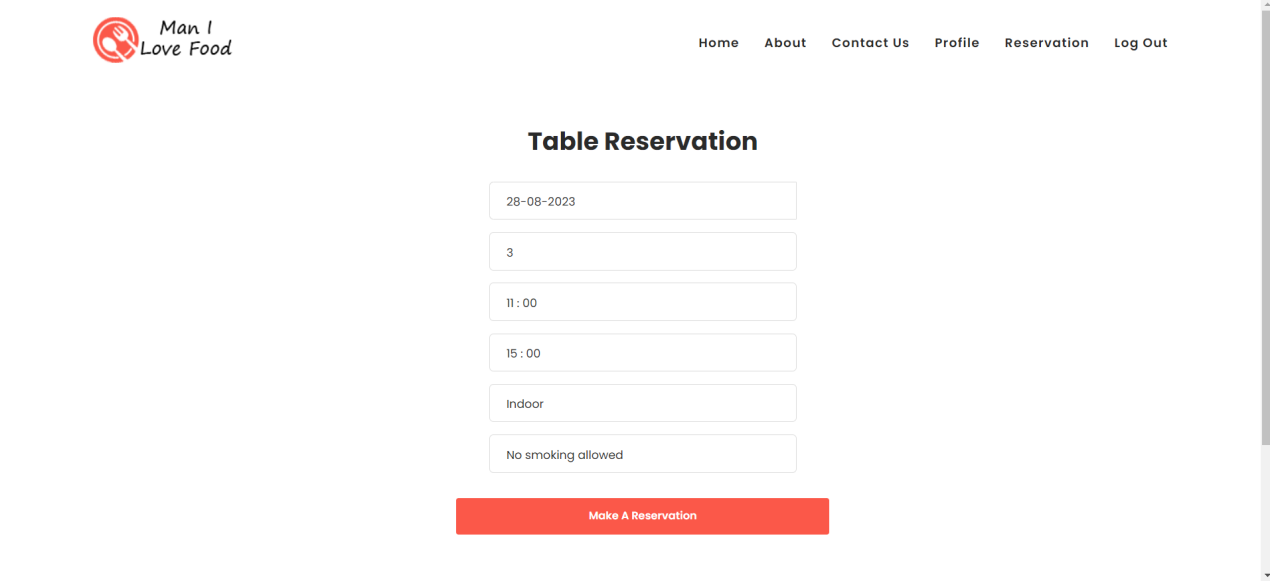
Nakon što se klikne na ***Reset password!*** dugme



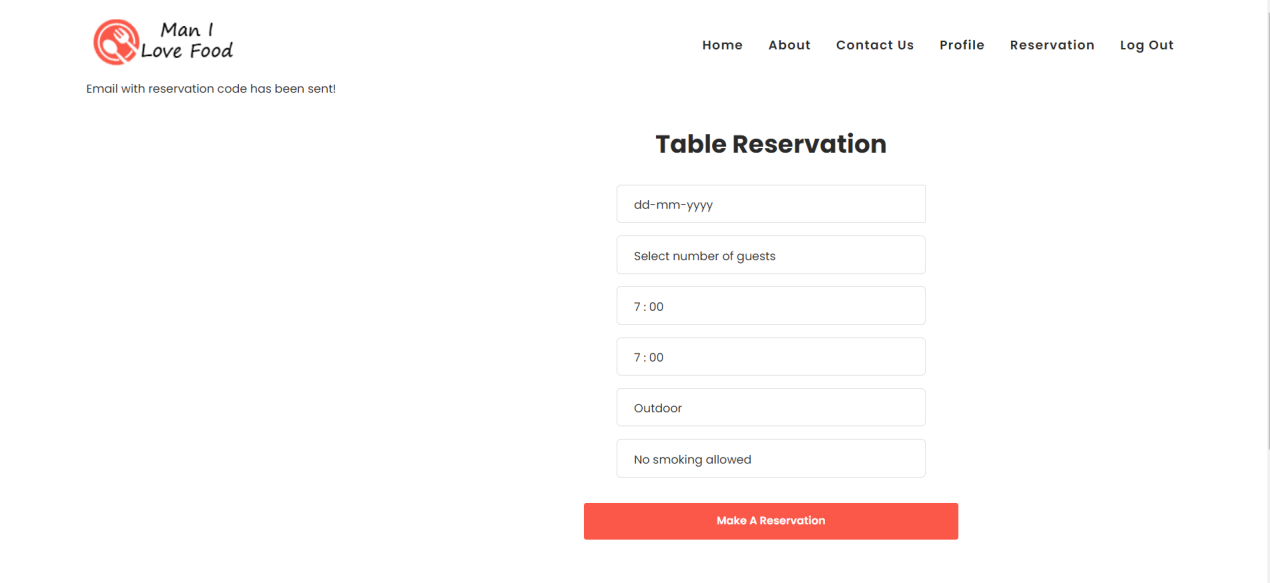
Nakon klika na link sa email-a u kome se nalzi i ***ip adresa*** korisnika koji je zatražio promenu lozinke.



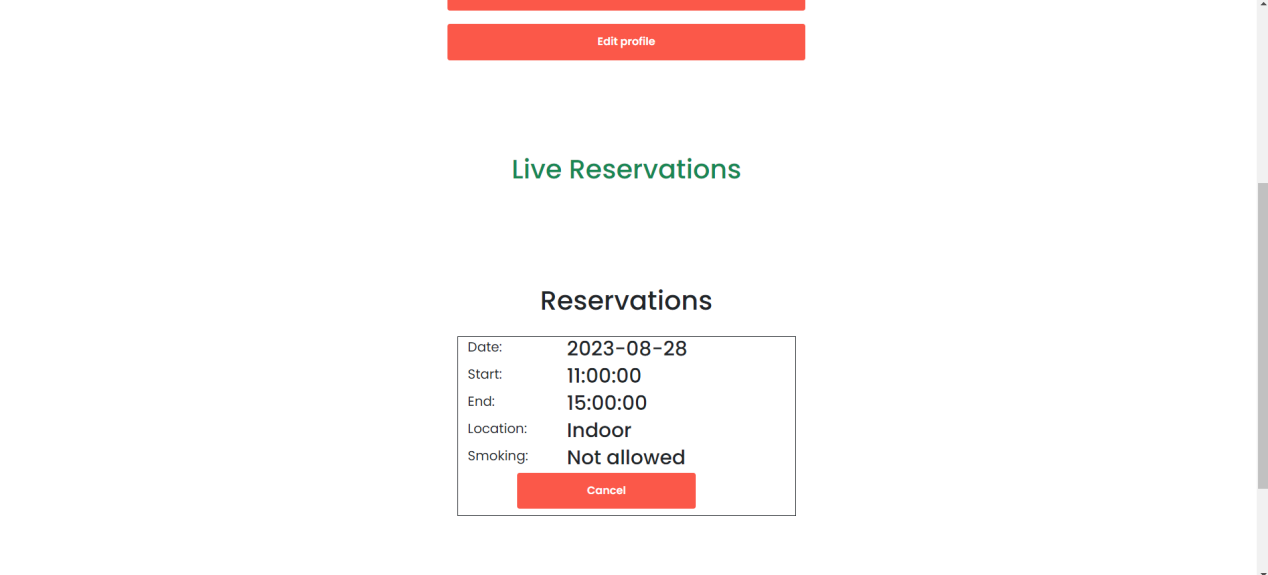
Rezervacija stola se izvršava nakon što su sva polja popunjena i nakon što se klike ***Make A Reservation*** dugme



Uspšno rezervisan sto

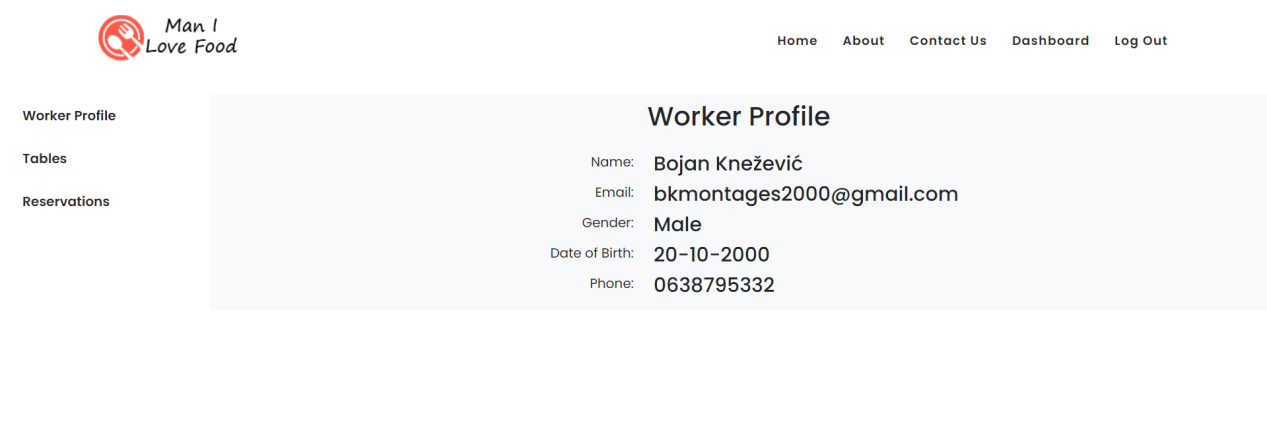


Rezervacije na profilu korisnika izgledaju ovako

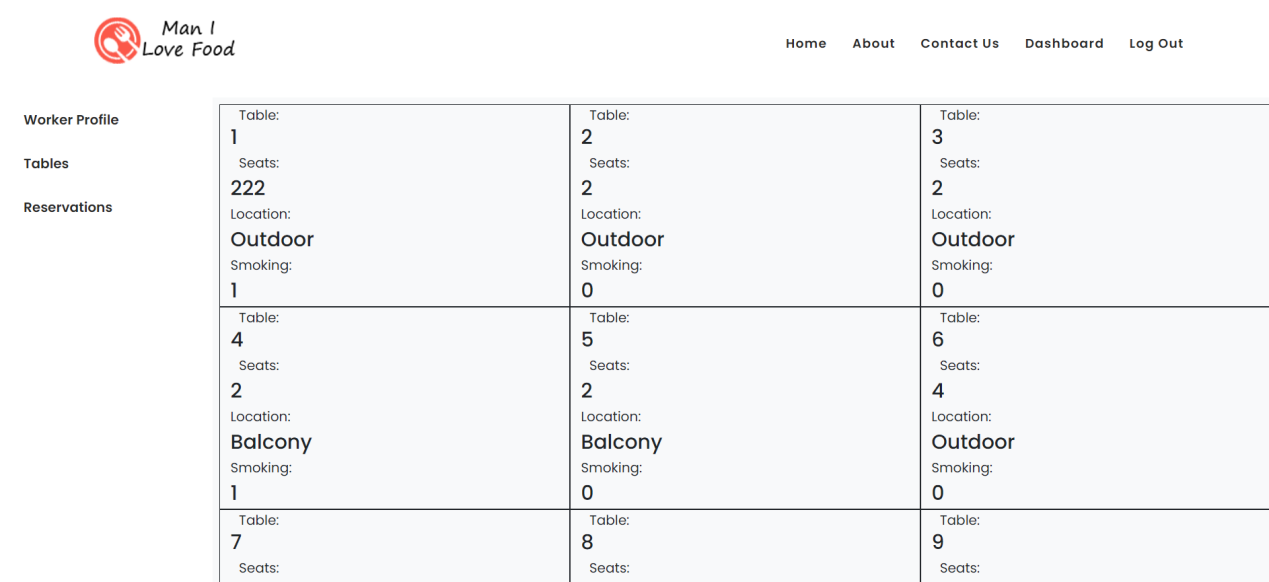


Otkazivanje rezervacije se vrši klikom na ***Cancel*** kod željene rezervacije. Kada dođe 4 časa do početka onda se rezervacija prebacuje u ***Live Reservations*** i ne može da se otkaže.

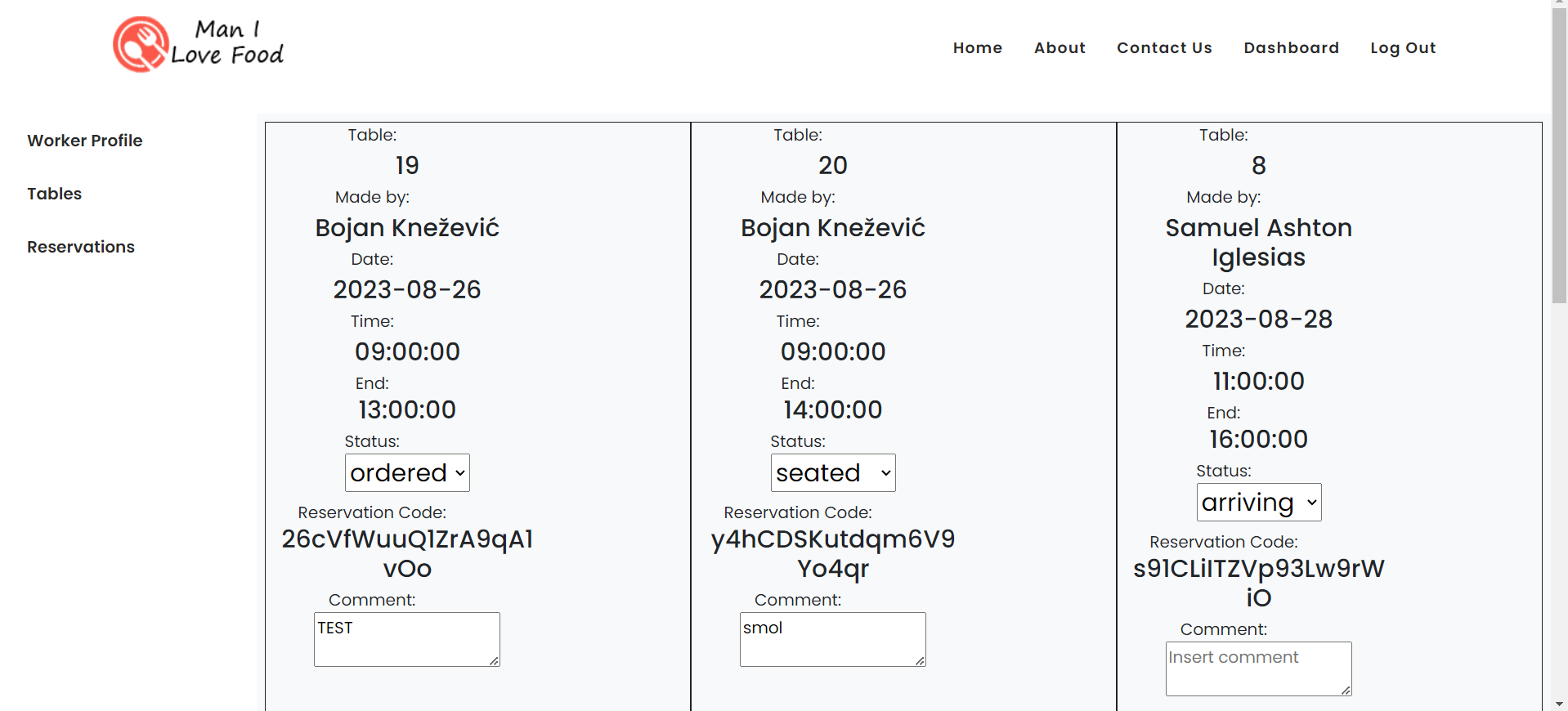
Prijavljivanje radnika se vrši putem ***Log in*** stranice. Kada se radnik prijavi umesto profila ima novi meni, u nastavku dashboard, preko kojeg može da vidi informacije o stolovima, menja status i komentar o rezervacijama kao i da vidi svoj profil.



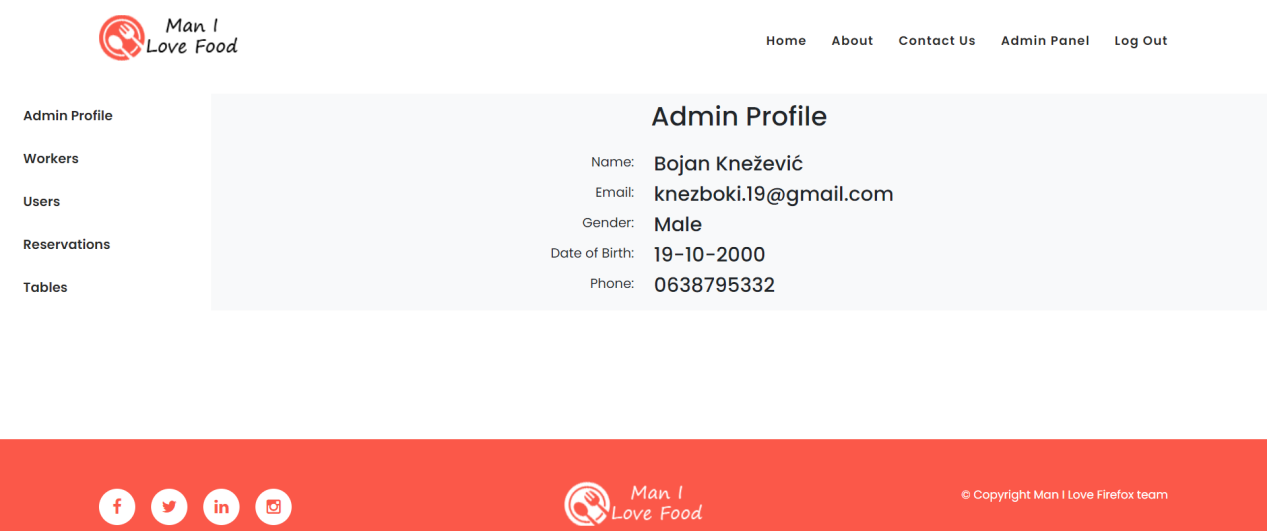
Informacije o stolovima



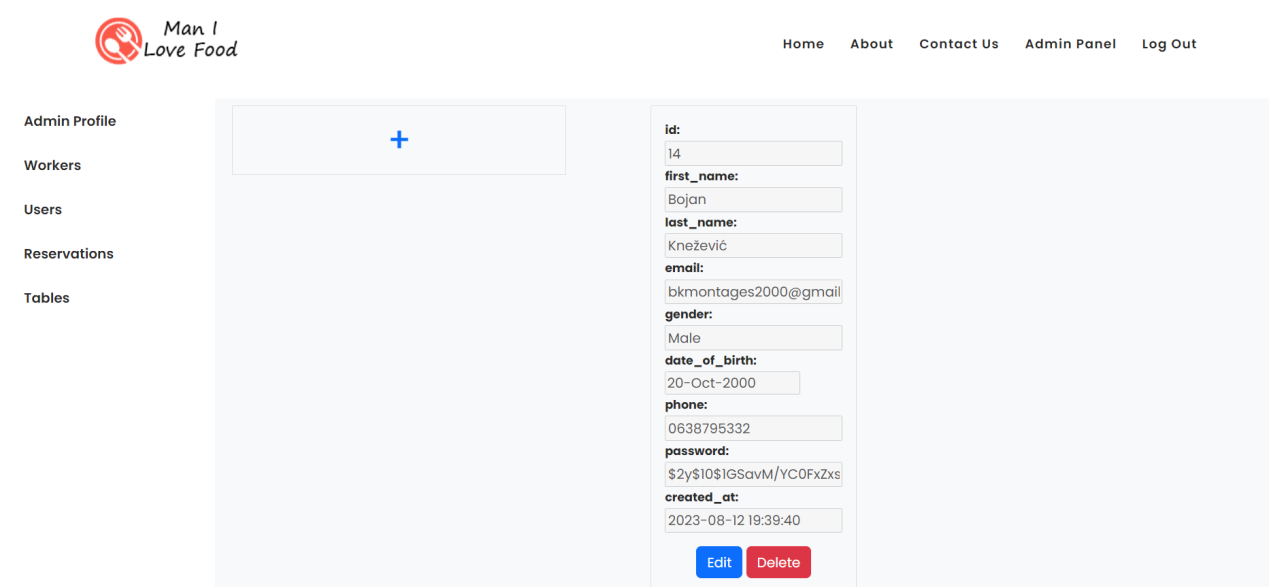
Rezervacije



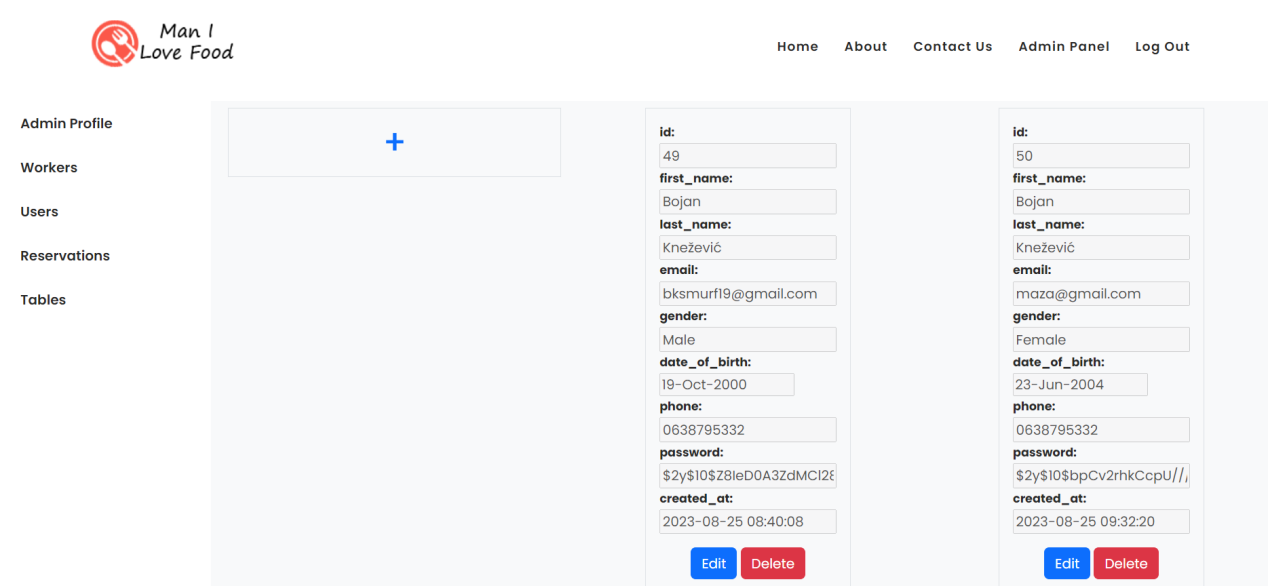
***Admin dashboard*** nakon sto se admin prijavi. Ovde može da vidi svoje detalje, listu radinika, korisnika, rezervacija i stolova. Pri čemu može da menja informacije radnika, korisnika i stolova klikom na ***Edit*** dugme ili briše klikom na ***Delete***. Može i da dodaje nove radnike i korisnike klikom na ***Plus(+)*** kod liste radnika odnosno korisnika.



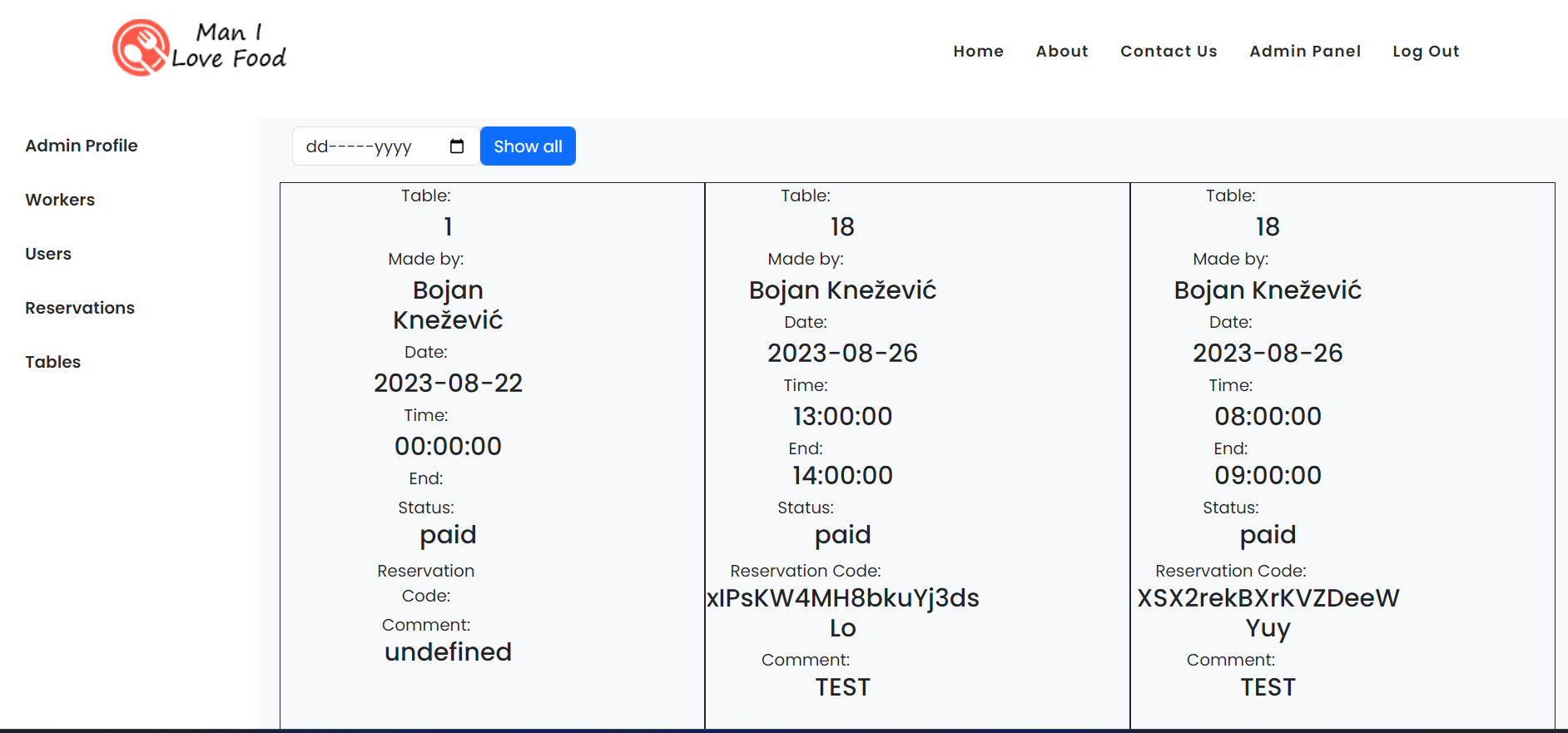
Lista radnika



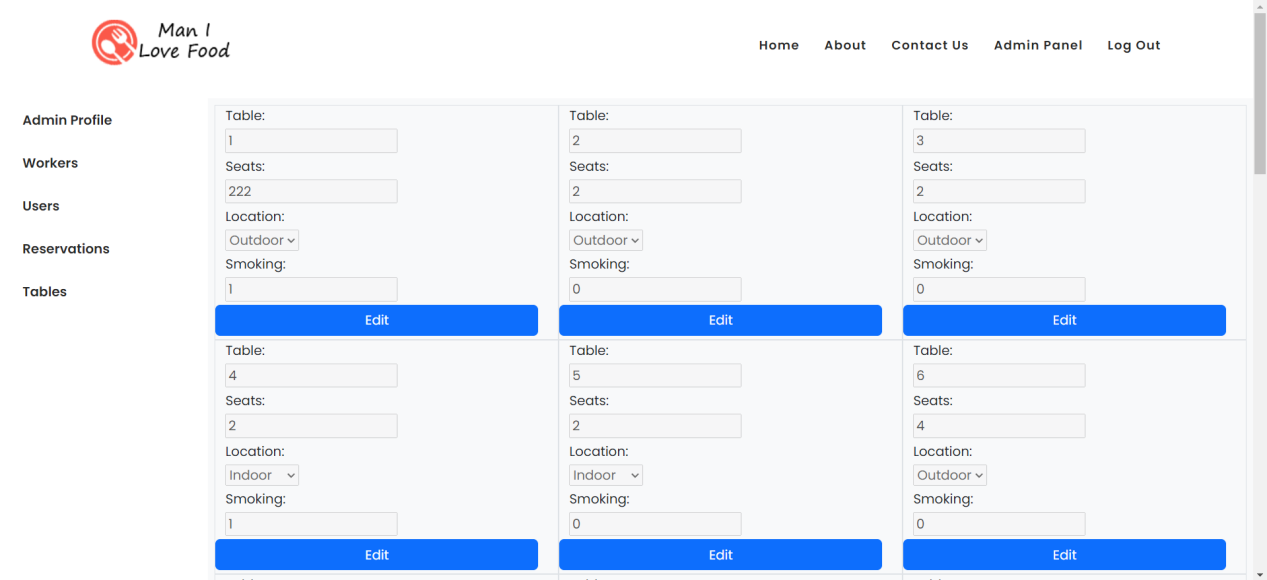
Lista korisnika



Lista rezervacija sa opcijom za prikaz svih ili samo nekih u zavisnosti od datuma nebitno da li su otkazane ili su gosti već platili i otišli.



Lista stolova gde mogu da se menjaju informacije o stolovima klikom na ***Edit*** dugme i sačuvaju izmene klikom na ***Save***.



I naravno svako od korisnika, radnika i admina može da se odjavi klikom na ***Log out*** u navigacionom meniju koje će uništiti sesiju i vratiti korisnika na ***log in*** stranicu.

# KORIŠĆENA LITERATURA

W1. <http://www.w3schools.com/>

W2. <http://www.jquery.com>

W3. <https://stackoverflow.com/>

W4. <https://people.vts.su.ac.rs/~chole/wp/primeri-peldak/>

W5. <https://mailtrap.io/home>

W6. <https://www.php.net/>