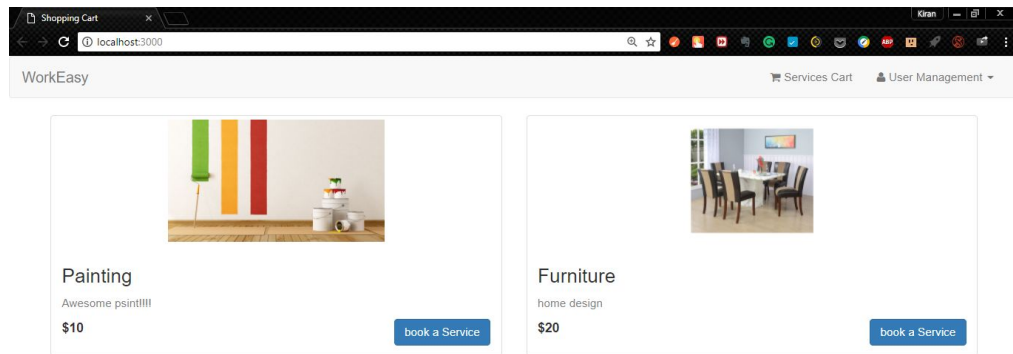


Home Page:

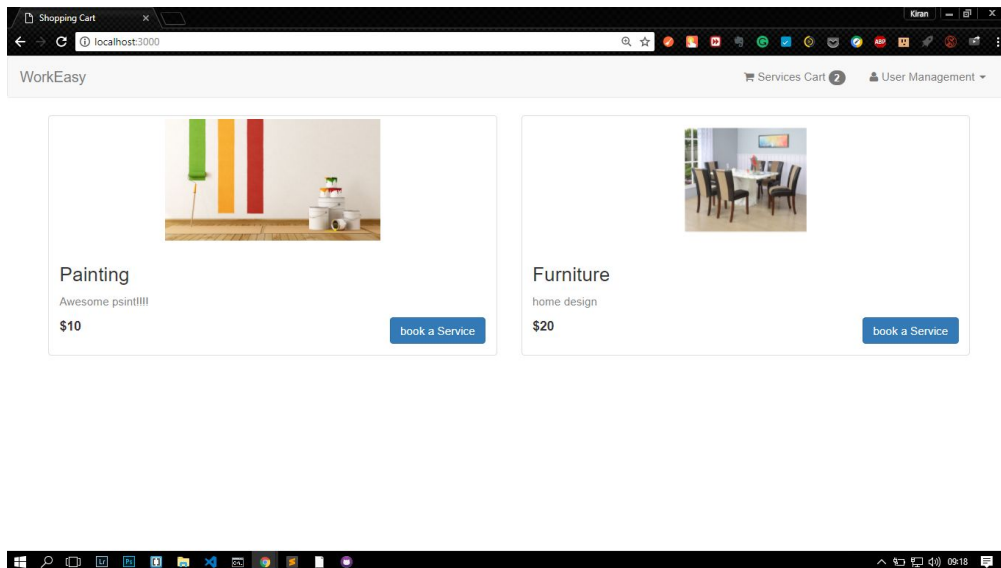
Starting page of application where user can choose the different services to book form.



Add To Cart Done:

User can click on book a service button.

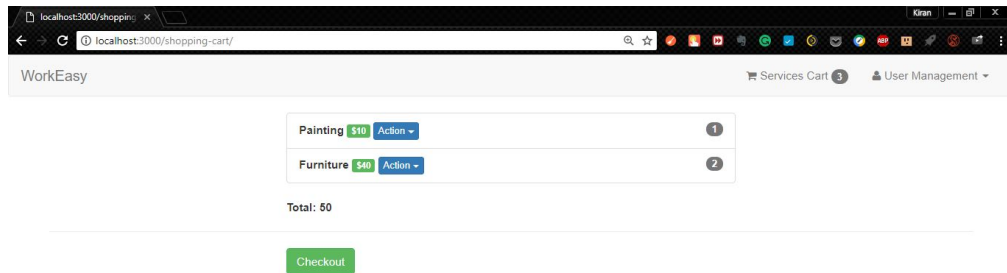
Then item will get add on service.



Cart Activity:

Services are added to cart to checkout

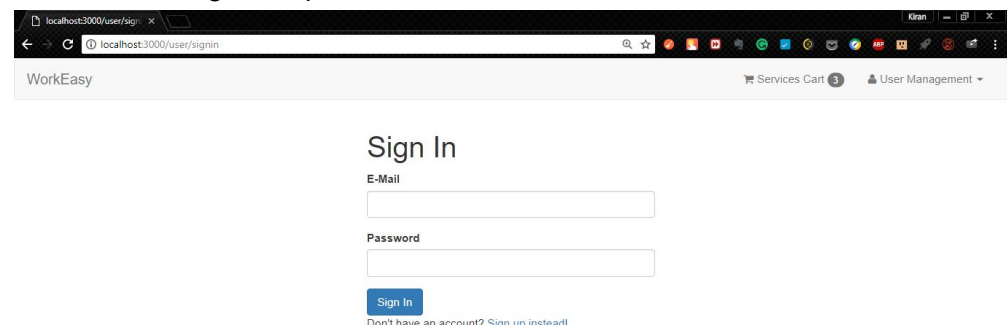
User can checkout o buy sevises.



Sign In page:

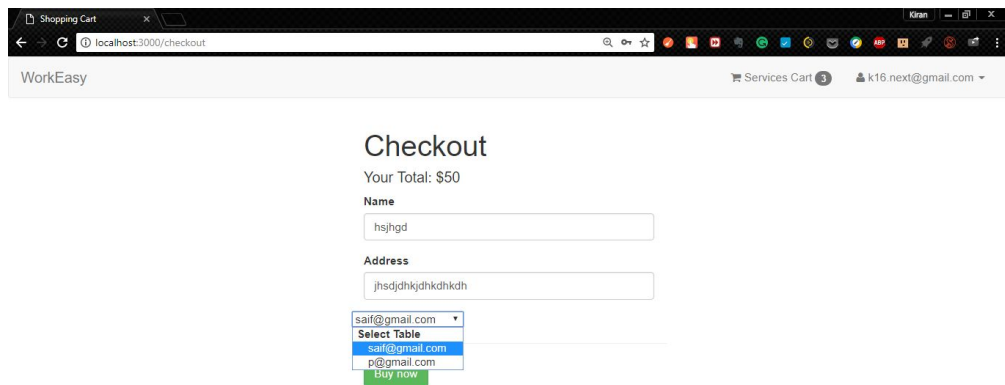
User and manager can sign In from here .

Forgot password if user forgot the password.



Checkout page :

Where user can checkout service and assign manager for that service.



Shopping Cart

localhost:3000/checkout

WorkEasy

Services Cart 3

k16.next@gmail.com

## Checkout

Your Total: \$50

Name

hsjhgd

Address

jhsdjhkdghkdghkdgh

saif@gmail.com

Select Table

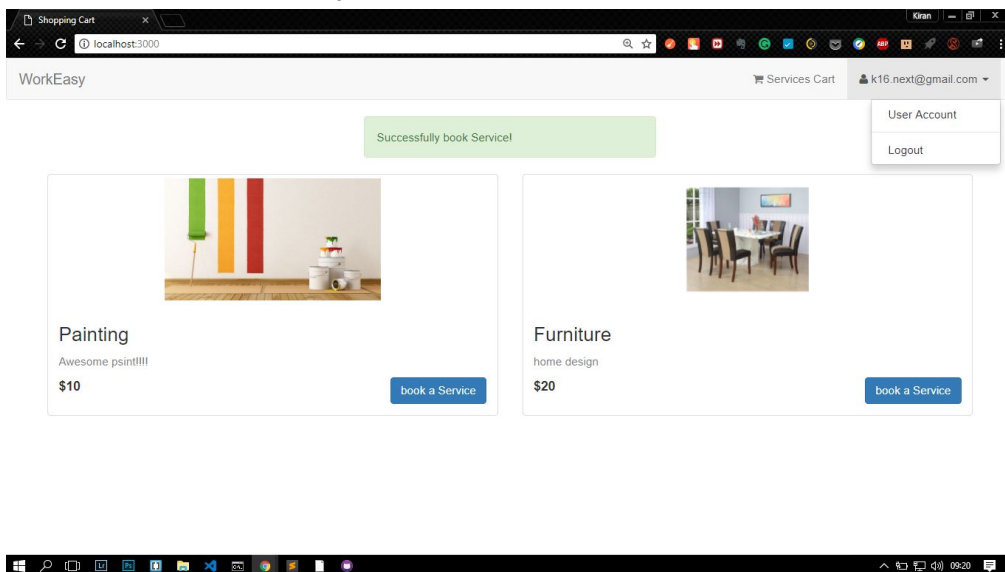
saif@gmail.com

p@gmail.com

buy now

Book service Success:

User will get popup when successfully book a service.



Shopping Cart

localhost:3000

WorkEasy

Services Cart

k16.next@gmail.com

Successfully book Service!

Painting

Awesome psint!!!!

\$10

book a Service

Furniture

home design

\$20

book a Service

User Account

Logout

## Index.js

```
var express = require('express');
var router = express.Router();
var Cart = require('../models/cart');
var Product = require('../models/product');
var Mman = require('../models/mman');
var Order = require('../models/order');
/* GET home page. */
router.get('/', function (req, res, next) {
  var successMsg = req.flash('success')[0];
  Product.find(function (err, docs) {
    var productChunks = [];
    var chunkSize = 3;
    for (var i = 0; i < docs.length; i += chunkSize) {
      productChunks.push(docs.slice(i, i + chunkSize));
    }
    res.render('shop/index', {title: 'Shopping Cart', products:
productChunks, successMsg: successMsg, noMessages: !successMsg});
  });
});
router.get('/add-to-cart/:id', function(req, res, next) {
  var productId = req.params.id;
  var cart = new Cart(req.session.cart ? req.session.cart : {});

  Product.findById(productId, function(err, product) {
    if (err) {
      return res.redirect('/');
    }
    cart.add(product, product.id);
    req.session.cart = cart;
    console.log(req.session.cart);
    res.redirect('/');
  });
});
router.get('/reduce/:id', function(req, res, next) {
  var productId = req.params.id;
  var cart = new Cart(req.session.cart ? req.session.cart : {});
  cart.reduceByOne(productId);
  req.session.cart = cart;
  res.redirect('/shopping-cart');
});
router.get('/remove/:id', function(req, res, next) {
  var productId = req.params.id;
  var cart = new Cart(req.session.cart ? req.session.cart : {});
  cart.removeItem(productId);
  req.session.cart = cart;
  res.redirect('/shopping-cart');
});
router.get('/shopping-cart', function(req, res, next) {
  if (!req.session.cart) {
    return res.render('shop/shopping-cart', {products: null});
  }
  var cart = new Cart(req.session.cart);
  res.render('shop/shopping-cart', {products: cart.generateArray(),
totalPrice: cart.totalPrice});});
```

```

router.get('/checkout', isLoggedIn, function(req, res, next) {
  if (!req.session.cart) {
    return res.redirect('/shopping-cart'); }
  var successMsg = req.flash('success')[0];
  Mman.find(function (err, docs) {
    var productChunks = [];
    var chunkSize = 3;
    for (var i = 0; i < docs.length; i += chunkSize) {
      productChunks.push(docs.slice(i, i + chunkSize));
    }
    var cart = new Cart(req.session.cart);
    var errMsg = req.flash('error')[0];
    res.render('shop/checkout', {title: 'Shopping Cart', products:
productChunks, successMsg: successMsg, noMessages: !successMsg, total:
cart.totalPrice, errMsg: errMsg, noError: !errMsg});
  });
  // var cart = new Cart(req.session.cart);
  // var errMsg = req.flash('error')[0];
  // res.render('shop/checkout', {total: cart.totalPrice, errMsg: errMsg,
noError: !errMsg});
});
router.post('/checkout', isLoggedIn, function(req, res, next) {
  if (!req.session.cart) {
    return res.redirect('/shopping-cart');
  }
  var cart = new Cart(req.session.cart);
  var order = new Order({
    user: req.user,
    cart: cart,
    address: req.body.address,
    name: req.body.name,
    mman: req.body.selectpicker,
  });
  order.save(function(err, result) {
    req.flash('success', 'Successfully book Service!');
    req.session.cart = null;
    res.redirect('/');
  });
});
module.exports = router;
function isLoggedIn(req, res, next) {
  if (req.isAuthenticated()) {
    return next();}
  req.session.oldUrl = req.url;
  res.redirect('/user/signin');
}

```

### **Passport.js**

```

var passport = require('passport');
var User = require('../models/user');
var Mman = require('../models/mman');
var LocalStrategy = require('passport-local').Strategy;
//var LocalStrategy2 = require('passport-local').Strategy;
                                passport.serializeUser(function
                                (user, done) {

```

```

        done(null, user);
    });

passport.deserializeUser(function
(user, done) {
    done(null, user);
});

passport.use('user.local.signin',
new LocalStrategy({
    usernameField: 'email',
    passwordField: 'password',
    passReqToCallback: true
}, function(req, email, password,
done) {
    req.checkBody('email', 'Invalid
email').notEmpty().isEmail();
    req.checkBody('password',
'Invalid password').notEmpty();
    var errors =
req.validationErrors();
    if (errors) {
        var messages = [];

errors.forEach(function(error) {

messages.push(error.msg);
        });
        return done(null, false,
req.flash('error', messages));
    }
    User.findOne({'email': email},
function (err, user) {
        if (err) {
            return done(err);
        }
        if (!user) {
            return done(null, false,
{message: 'No user found.'});
        }
        if
(!user.validatePassword(password)) {
            return done(null, false,
{message: 'Wrong password.'});
        }
        return done(null, user);
    });
}));

```