**King Fahd University of Petroleum & Minerals**
**College of Computer Sciences and Engineering**

# SWE 363

# Project Phase 3: READy Bookstore

**by**

**Mostafa Othman 201943910**

**Mohammad Aldosare 201924770**

**Mohammed Alkhnani 201954190**

**Turki Alzahrani 201936550**

**For**

**Dr. Alaaddin Sabour & Dr. MD MUSFIQUE ANWAR**

**Web Engineering and Development**

**SWE 363**

# Description:

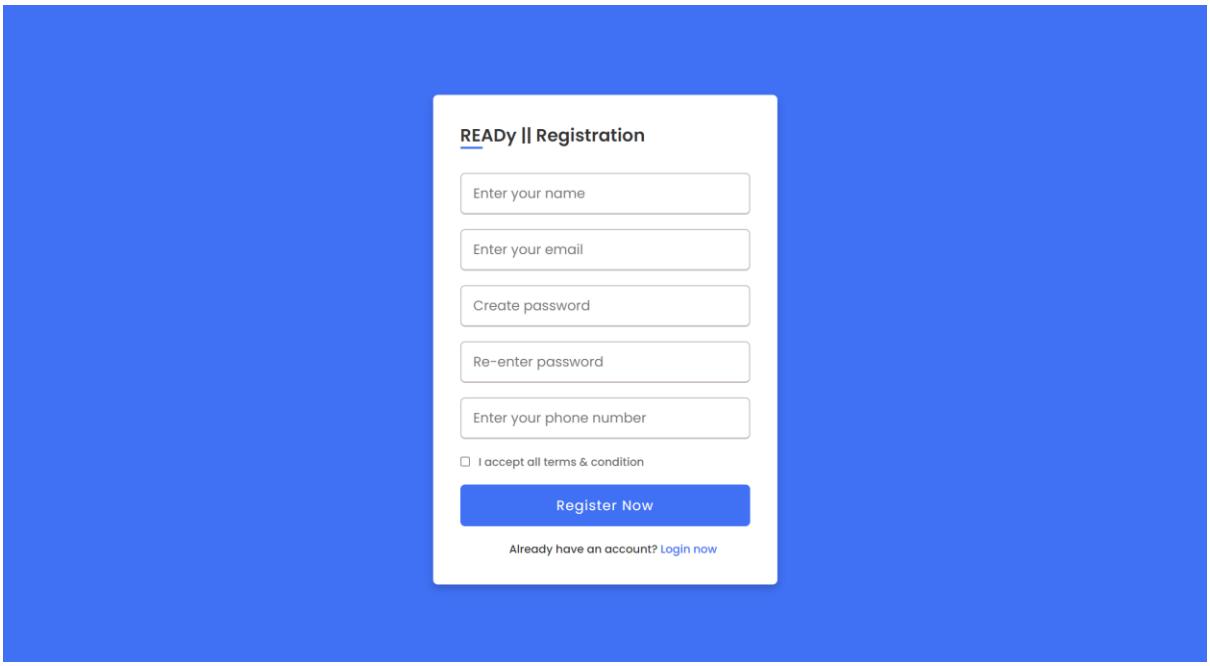
READy is a Platform That Encourages People to Buy and Read Books. The Name Combines Two of Our Main Focuses, "Read" Which Emphasizes the Primary Aim of Our Platform, Providing Readable Materials Via Whatever Means That Are Most Convenient for Our Customers. Convenience Leads Us to The Other Focus, "Ready" We Believe That the Best Way to Acquire Customers and Retain Their Trust Is by Providing Them With An Outstanding And Convenient Experience That Makes Them Want To Return.

# Technologies and Tools:

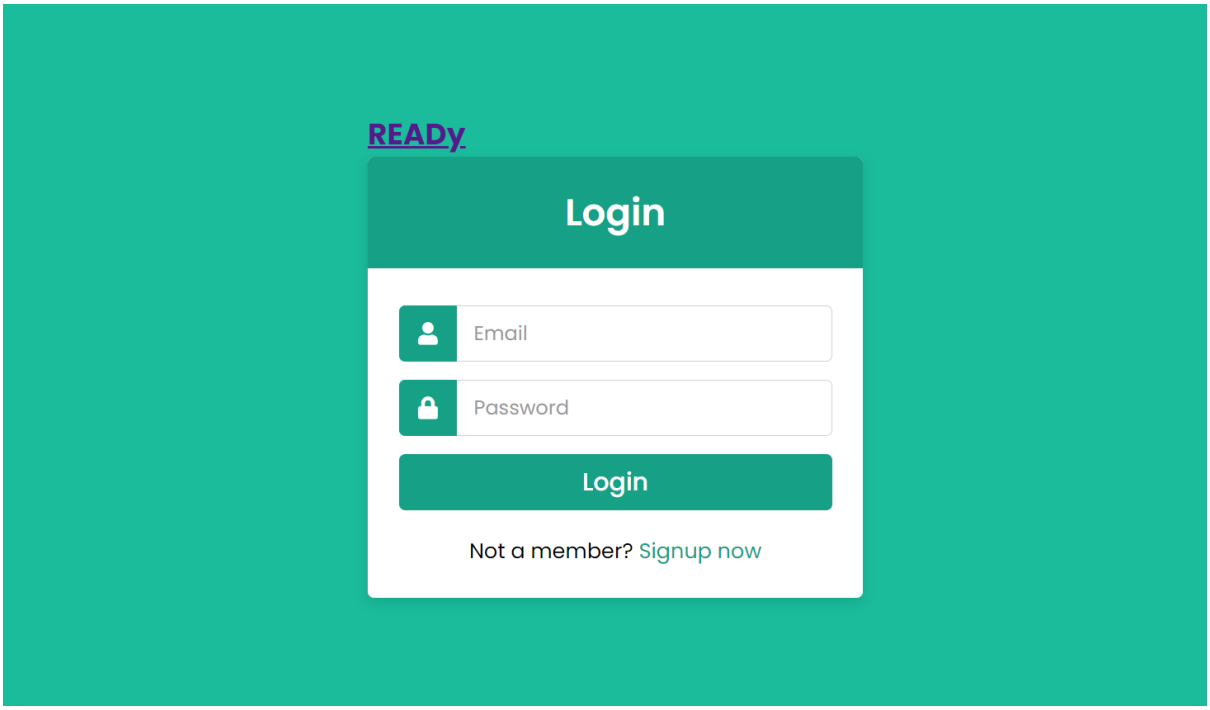
- HTML
- CSS
- Javascript
- Node.js, Express
- MongoDB, Mongoose
- Passport JS
- Bcrypt
- EJS
- Github

# GUI Screen Shots:

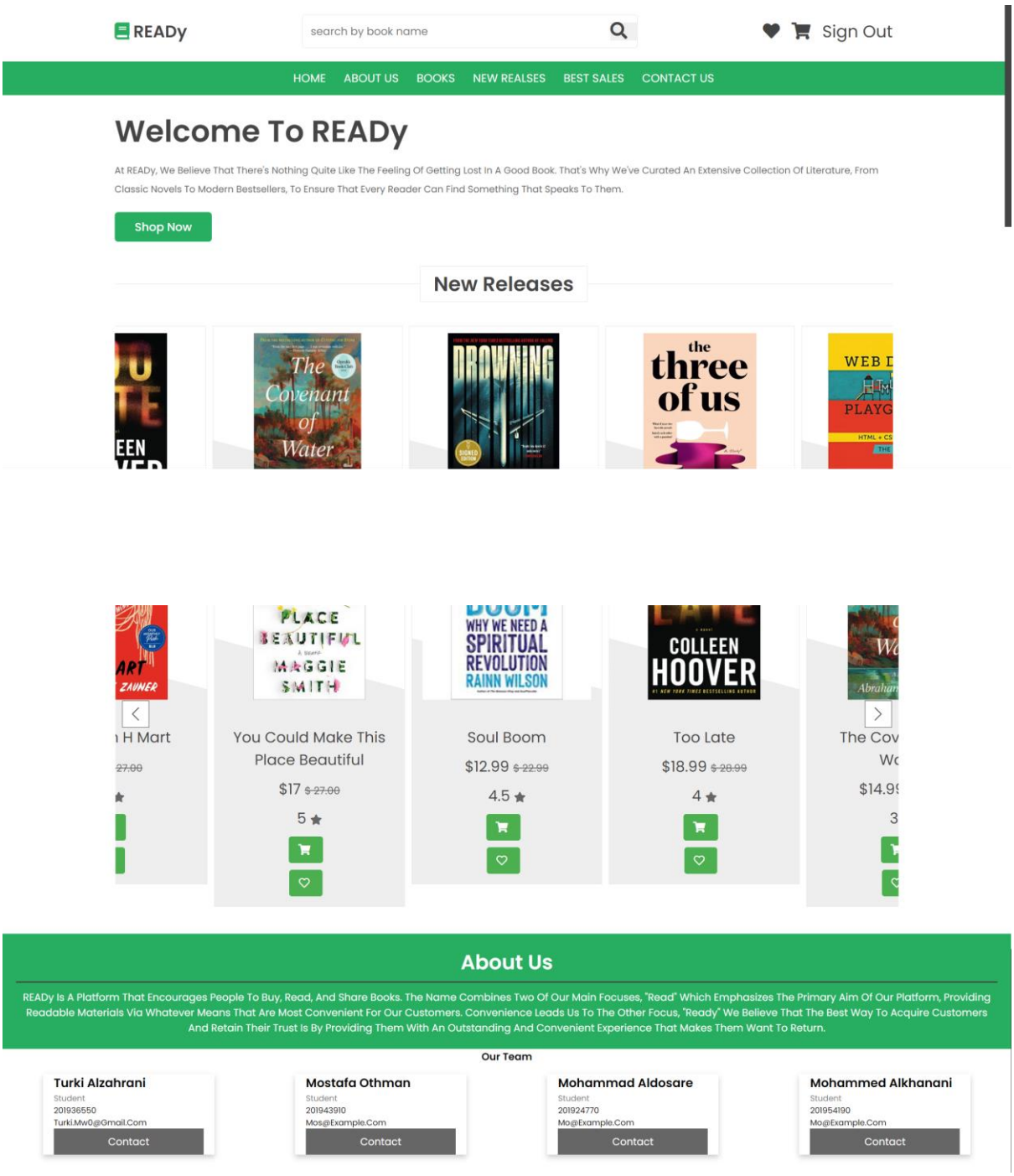
Registration:



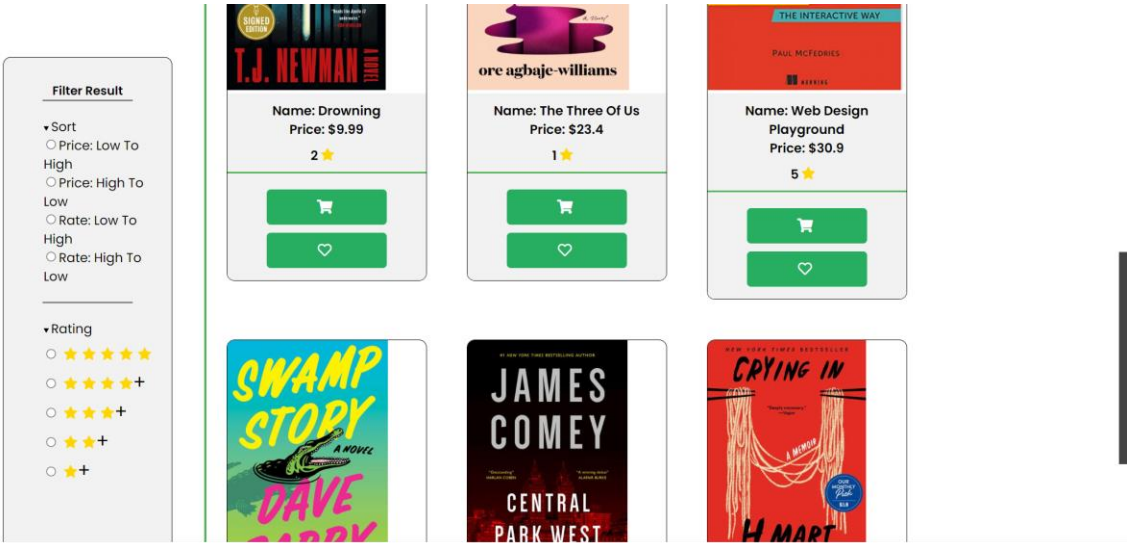
log in:



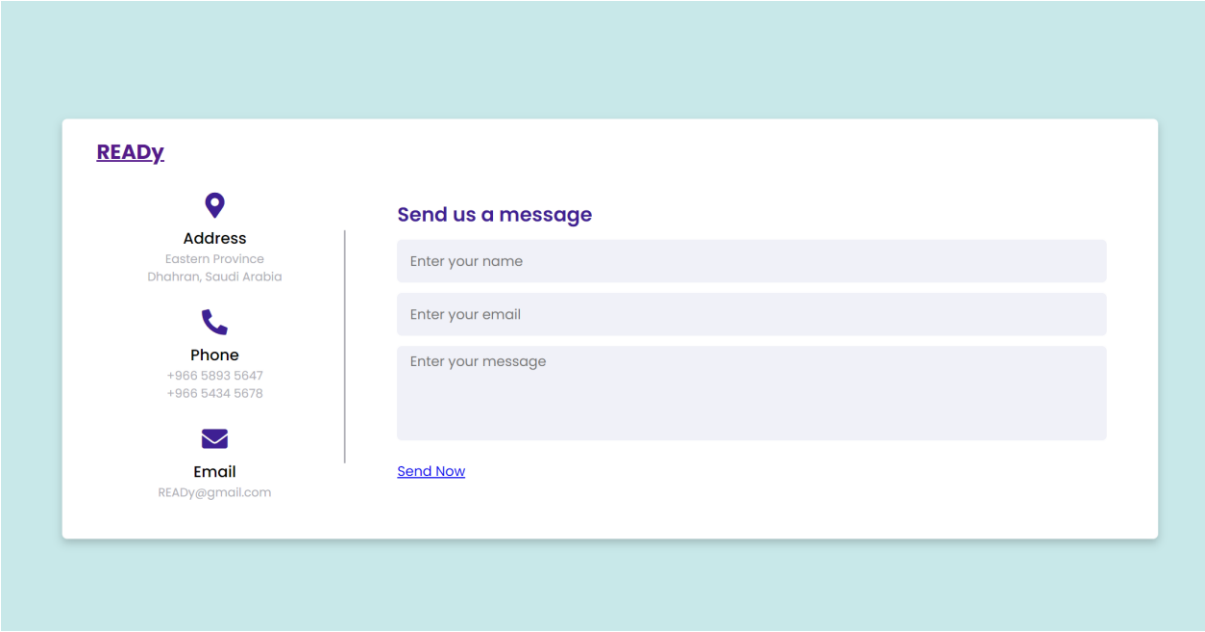

Home page:


READy
search by book name
Sign Out
HOME ABOUT US BOOKS NEW REALSES BEST SALES CONTACT US
Welcome To READy
At READy, We Believe That There's Nothing Quite Like The Feeling Of Getting Lost In A Good Book. That's Why We've Curated An Extensive Collection Of Literature, From Classic Novels To Modern Bestsellers, To Ensure That Every Reader Can Find Something That Speaks To Them.
Shop Now
New Releases
The Covenant of Water
DROWNING
the three of us
You Could Make This Place Beautiful
$17 $27.00
5
Soul Boom
$12.99 $22.99
4.5
Too Late
$18.99 $28.99
4
About Us
READy Is A Platform That Encourages People To Buy, Read, And Share Books. The Name Combines Two Of Our Main Focuses, "Read" Which Emphasizes The Primary Aim Of Our Platform, Providing Readable Materials Via Whatever Means That Are Most Convenient For Our Customers. Convenience Leads Us To The Other Focus, "Ready" We Believe That The Best Way To Acquire Customers And Retain Their Trust Is By Providing Them With An Outstanding And Convenient Experience That Makes Them Want To Return.
Our Team
Turki Alzahrani
Student
201936550
Turki.Mw0@Gmail.Com
Contact
Mostafa Othman
Student
201943910
Mos@Example.Com
Contact
Mohammad Aldosare
Student
201924770
Mo@Example.Com
Contact
Mohammed Alkhanani
Student
201954190
Mo@Example.Com
Contact

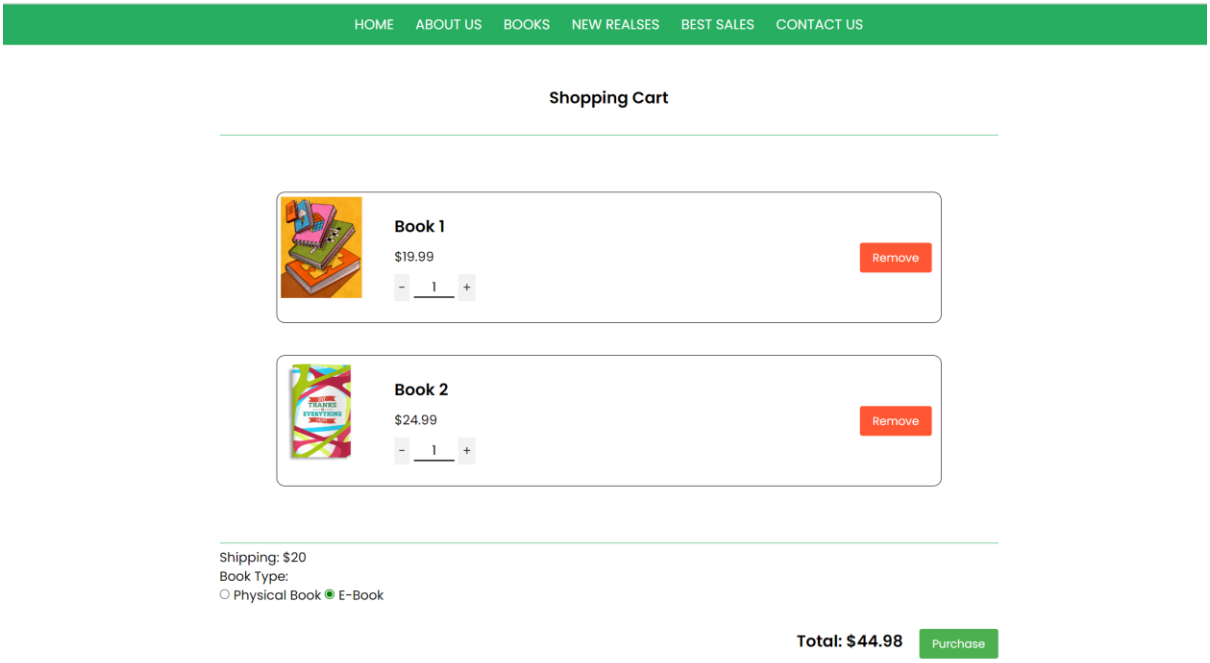
Books page:



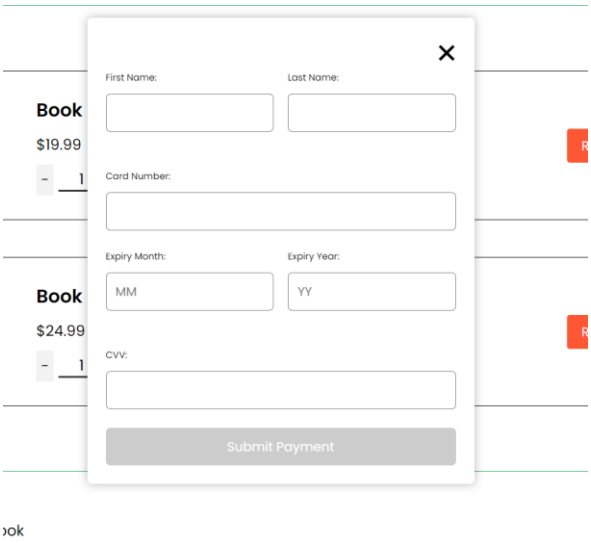

Contact us page:



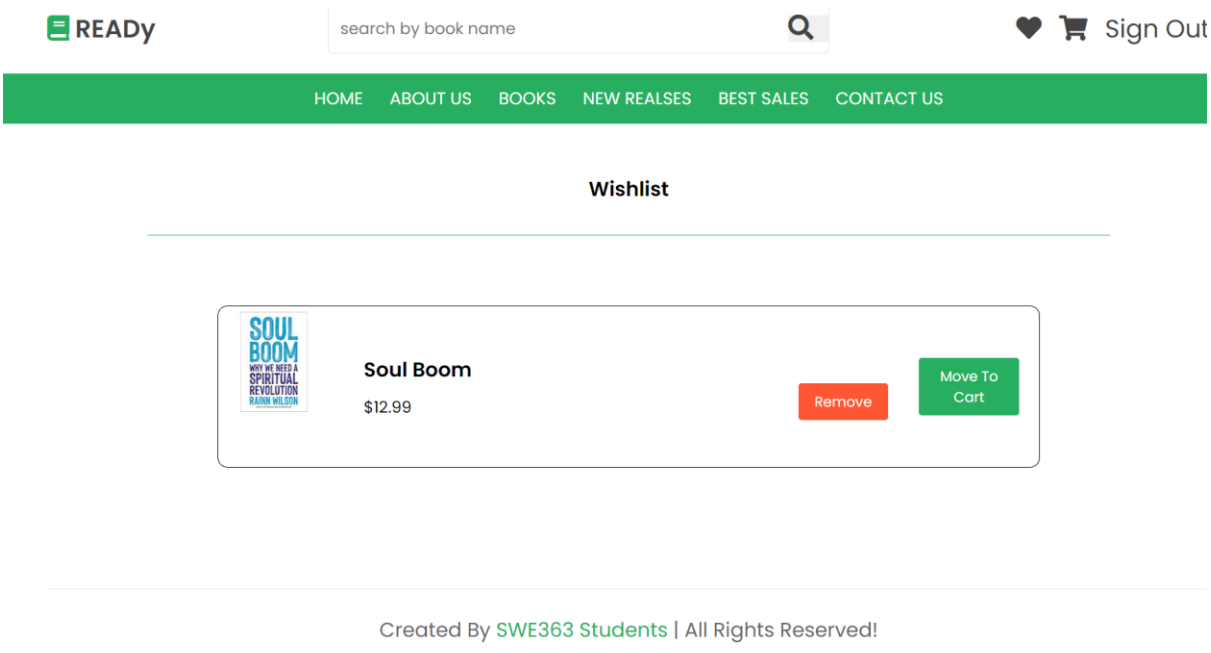

Shopping cart page:



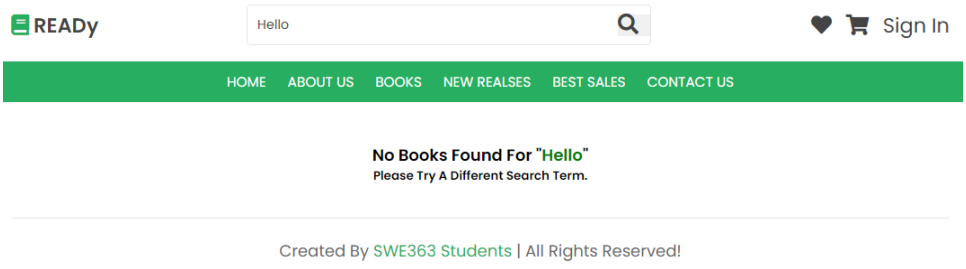

Payment window:



Wishlist page:




No_book_found page




# Source Code Snapshots:

This is the user model for the users collection:

```
models > JS book.js > ...
const mongoose = require('mongoose')

const bookSchema = new mongoose.Schema({
    nameOfBook: String,
    priceOfBook: Number,
    rateOfBook: Object,
    descOfBook: String,
    imgSrc: String,
    authorOfBook: String,
    bookFormat: String,
    genreOfBook: String,
    pageNumOfBook: String,
    index: Number,
    totalRate: Number,
})



module.exports = Book = mongoose.model("books", bookSchema)
```

This is the book model for the books collections:

```
models > JS user.js > [e] userSchema
const mongoose = require('mongoose')

const bookSchema = new mongoose.Schema({
    nameOfBook: String,
    priceOfBook: Number,
    rateOfBook: Number,
    imgSrc: String,
})

const userSchema = new mongoose.Schema({
    name: String,
    phone: String,
    email: {
        type: String,
        required: true,
    },
    password: {
        type: String,
        required: true,
    },
    orders: [bookSchema],
    wishlist: [bookSchema],
})

module.exports = User = mongoose.model("users", userSchema)
```

If the user is logged in it will display top-right Sign out, else it will display Sign in

```
views > Book.ejs > html > body > header.header > section.header-1 > div.icons > ? > ? > ? > ? > ?
<form action="/search" method="get" class="search-form">
  <input type="search" name="keyword" placeholder="search by book name" id="search-box">
  <button type="submit">
    <label for="search-box" class="fas fa-search"></label>
  </button>
</form>

<div class="icons">
    <div id="search-btn" class="fas fa-search"></div>
    <a href="/wishlist" class="fas fa-heart"></a>
    <a href="/cart" class="fas fa-shopping-cart"></a>
    <% try{ %>
      <% if(user.email != undefined) {%>
        <a id="login-btn" class="#" href="/logout">
          Sign Out
        </a> <!-- Because of ejs these has to be changed... from login.ejs to just login -->
    <% } else {%>
      <a id="login-btn" class="#" href="/login">
        Sign In
      </a>
      <% }} catch { %>
        <a id="login-btn" class="#" href="/login">
          Sign In
        </a>
        <% } %>
</div>
</div>
```

This is how we are getting the data of the book from the server-side using EJS

```
<div class="details">

  <ul class="spec-table">
    <li class="table-row">
      <div class="col col-1" data-label="Job Id">Author</div>
      <div class="col col-2" data-label="Customer Name">&lt;%= book.authorOfBook %&gt;</div>

    </li>
    <li class="table-row">
      <div class="col col-1" data-label="Job Id">Number of pages</div>
      <div class="col col-2" data-label="Customer Name">&lt;%= book.pageNumOfBook %&gt;</div>

    </li>
    <li class="table-row">
      <div class="col col-1" data-label="Job Id">Genre</div>
      <div class="col col-2" data-label="Customer Name">&lt;%= book.genreOfBook %&gt;</div>

    </li>
    <li class="table-row">
      <div class="col col-1" data-label="Job Id">book format</div>
      <div class="col col-2" data-label="Customer Name">&lt;%= book.bookFormat %&gt;</div>

    </li>
    <li class="table-row">
      <div class="col col-1" data-label="Job Id">ISBN</div>
      <div class="col col-2" data-label="Customer Name">&lt;%= book._id %&gt;</div>
```

If the user is not logged in and clicks on add to cart/wishlist he/she will be directed to the login page

```
<% try{ %>
  <% if(user.email != undefined) {%>
<form action="/cart" method="post">
  <input type="hidden" name="user_email" value="<%=user.email%>" id="">
  <input type="hidden" name="book_name" value="<%= book.nameOfBook %>" id="">
<button type="submit" class="btn cart-btn"><i class="fas fa-shopping-cart"></i></button>
</form>
<form class="bb1" action="/wishlist" method="post">
  <input type="hidden" name="user_email" value="<%=user.email%>" id="">
  <input type="hidden" name="book_name" value="<%= book.nameOfBook %>" id="">
<button  type="submit" class="btn wishlist-btn"><i class="far fa-heart"></i></button>
</form>
<% } else {%> <form action="/login" method="get">
  <button type="submit" class="btn cart-btn" ><i class="fas fa-shopping-cart" ></i></button>
</form>
<form action="/login" method="get">
    <button type="submit" class="btn wishlist-btn"><i class="far fa-heart"></i></button>
</form>
<% }} catch { %>
  <form action="/login" method="get">
  <button type="submit" class="btn cart-btn" ><i class="fas fa-shopping-cart" ></i></button>
</form>
<form action="/login" method="get">
    <button type="submit" class="btn wishlist-btn"><i class="far fa-heart"></i></button>
</form>
<% } %>
```

This is how we get the book rating from a registered account in book.ejs

```
<script>
document.addEventListener('DOMContentLoaded', function() {
  const ratingForm = document.querySelector('#rating-form');
  const starElements = ratingForm.querySelectorAll('.rating-stars i');
  starElements.forEach(function(starElement) {
    starElement.addEventListener('click', function() {
      const ratingValue = starElement.getAttribute('data-rating');
      const ratingInput = ratingForm.querySelector('.rating-value');
      ratingInput.value = ratingValue;
      ratingForm.submit();
    });
  });
});
</script>
```

This is how we handle the sorting request from the user in books.ejs

```
<script>
  searchForm = document.querySelector('.search-form')

document.querySelector('#search-btn').onclick = () => {
  searchForm.classList.toggle('active')
 }
      // get the form element
      let form1 = document.getElementById('sort-form');

     // add an event listener to the radio buttons to listen for changes in the selected option
     let sortRadioButtons1 = document.getElementsByName('sort');
     for (let i = 0; i < sortRadioButtons1.length; i++) {
         sortRadioButtons1[i].addEventListener('change', function() {
             // submit the form automatically when a radio button is selected
             form1.submit();
         });
     }

     let form2 = document.getElementById('rate-form');

     // add an event listener to the radio buttons to listen for changes in the selected option
     let sortRadioButtons2 = document.getElementsByName('rate');
     for (let i = 0; i < sortRadioButtons2.length; i++) {
         sortRadioButtons2[i].addEventListener('change', function() {
             // submit the form automatically when a radio button is selected
             form2.submit();
         });
     }
```

Getting new account information through a post method in register.ejs

```
    <form action="/register" method="POST">
      <div class="input-box">
        <input type="text" id="name" name="name" placeholder="Enter your name" required>
      </div>
      <div class="input-box">
        <input type="email" id="email" name="email" placeholder="Enter your email" required>
      </div>
      <div class="input-box">
        <input type="password" id="password" name="password" placeholder="Create password" required>
      </div>
      <div class="input-box">
        <input type="password" id="confirm-password" name="confirm-password" placeholder="Re-enter password" requ
      </div>
      <div class="input-box">
        <input type="number" id="phone" name="phone" placeholder="Enter your phone number" required>
      </div>
      <div class="policy">
        <input type="checkbox" required>
        <h3>I accept all terms & condition</h3>
      </div>
      <div class="input-box button">
        <input type="Submit" value="Register Now">
      </div>
      <div class="text">
        <h3>Already have an account? <a href="/login">Login now</a></h3>
      </div>
      <% if (messages.error) { %>
        <p style="color: red"><%- messages.error -%></p>
      <% } %>
    </form>
  </div>
```

Password validation in register.ejs

```
<script>
  // Add client-side validation for password and confirm-password fields
  const passwordField = document.getElementById('password');
  const confirmPasswordField = document.getElementById('confirm-password');

  function validatePasswords() {
    if (passwordField.value !== confirmPasswordField.value) {
      confirmPasswordField.setCustomValidity("Passwords do not match.");
    } else {
      confirmPasswordField.setCustomValidity("");
    }
  }

  passwordField.addEventListener("change", validatePasswords);
  confirmPasswordField.addEventListener("keyup", validatePasswords);
</script>
</body>
```

Authentication using the Passport JS

```
function initilaize(passport) {
    const authenticateUser = async (email, password, done) => {
        const user = await User.findOne({email: email})
        if(user == null) {
            return done(null, false, {message: "no user with that email"})
        }
        try {
            if (await bcrypt.compare(password, user.password)) {
                return done(null, user)
            } else {
                return done(null, false, {message: "Password is incorrect"})
            }
        } catch(e) {
            return done(e)
        }
    }
    passport.use(new LocalStrategy({usernameField: 'email'}, authenticateUser))

    passport.serializeUser((user, done) => {
        return done(null, user._id)
    })
    passport.deserializeUser((id, done) => {
        User.findById(id)
          .then(user => {
            done(null, user);
          })
          .catch(err => {
            done(err);
          });
      });
}
module.exports = initilaize
```

This is how we connect to the database

```
uri = 'mongodb+srv://admin:admin@cluster0.moh5hyj.mongodb.net/?retryWrites=true&w=majority'
mongoose.connect(uri, {useNewUrlParser: true});

initilaizePassport(passport, async (email) => {
    const result = await User.find({email: email})
    return result[0];
})

const app = express()
app.use(express.static('public'));
app.use(express.urlencoded({extended: false}));
app.use(flash())
app.use(session({
    secret: 'secret',
    resave: false,
    saveUninitialized: false
}))

app.use(passport.initialize())
app.use(passport.session())

app.set('view engine', 'ejs');

function ensureAuthenticated(req, res, next) {
    if (req.isAuthenticated()) {
      return next();
    } else {
      res.redirect('/login');
    }
  }
```

This is an example of how we handle get and post methods

```
app.get('/register', (req, res) => {
    res.render('register.ejs')
})
app.post('/register', async (req, res) => {
    const user = await User.findOne({
        email: req.body.email
      })

    try {
        const hashedPassword = await bcrypt.hash(req.body.password, 10) //hashing the password
        if (user) {
            req.flash('error', 'Sorry, that name is taken. Maybe you need to <a href="/login">login</a>?');
            res.redirect('/register');
          } else if (req.body.email == "" || req.body.password == "") {
            req.flash('error', 'Please fill out all the fields.');
            res.redirect('/register');
          } else {
            const user = new User({
                name: req.body.name,
                phone: req.body.phone,
                email: req.body.email,
                password: hashedPassword,
                orders: [],
                wishlist: []
                })
                await user.save() // saving the user to DB

                console.log("user add to the database")
                res.redirect('/login')
          }

    } catch (error){
        console.error(error.message)
        res.redirect('/register')
    }
})
```

Note: Server.js contains a lot of important source code, this is just an example.

# Open-source code reusability:

We used font packages like:

- https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css
- https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css

And we used a package for the swiper:

- https://unpkg.com/swiper@7/swiper-bundle.min.css

# Conclusion:

*Learnings from the project:*

- We learned to apply our knowledge of frontend, backend, and server-side programming to develop a complete website that is neat, smooth, and responsive.
- We learned that planning and organization are critical especially when working as a group on complex projects where many tasks can be overlapping.
- We learned that collaboration is essential each team member brings unique skills and expertise to the project, and it is important to work together to leverage these strengths effectively.
- We learned a lot of web technologies like passport js for authentication, bcrypt to encrypt the passwords that are stored in the database.
- We learned how to solve errors that we faced during the development of the project.

*Limitations of your work:*

- The website does not have the ability to process real life transactions, for example, when the user advances to enter his/her credit card details and then press on purchase. Nothing happens
- The contact us page and the about us section in the home page is not functional as the email used in there does not belong to any of the team members.
- The best sales and new releases sections on our website are not based on actual sales or releases data.

*What would you do differently:*

- We would try to better understand our requirements before rushing to start developing.
- We would try to learn git before starting the project for better handling of our source code.
- We would try to better organize our work distribution.
- We would have learned an entire frontend framework like react or angular and then apply our knowledge to the project.
- We would have not used mongoDB because sometimes it is slow internet wise, also mongoDB does not work on the KFUPM internet (IP related issue)