Image Search Generator (PICTOPIA)

Abstract :

Pictopia is a web-based application designed to simplify the process of searching, discovering, and managing high-quality images sourced from the **Unsplash API**. The system allows users to register, log in, and securely manage their favorite images. Upon authentication, users can search for images using keywords, view results in a responsive grid layout, and save or remove images from their favorites list.

Built using the **MEAN stack (MongoDB, Express.js, Angular, Node.js)**, Pictopia ensures efficient communication between its frontend and backend components. MongoDB securely stores user data and favorite images, while JWT (JSON Web Token) manages secure access to protected routes.

By offering a smooth and user-friendly interface for managing and discovering images, this system improves the user experience while utilizing contemporary web technologies for security and scalability.

Introduction :

In today's digital landscape, visual content is a cornerstone of online interaction, yet organizing and managing image collections can be cumbersome. Pictopia addresses this challenge by offering a streamlined platform for users to search, discover, and curate images from the Unsplash API.

Developed using the **MEAN stack**—MongoDB, Express.js, Angular, and Node.js—Pictopia ensures a responsive and secure user experience. The application features user authentication via JWT, enabling personalized functionalities such as saving favorite images. Users can search

for images using keywords, view results in a grid layout, and manage their favorites with ease. The system not only optimizes image discovery but also provides a secure and intuitive interface, making it an ideal solution for users seeking a hassle-free way to organize visual content.

Pictopia improves efficiency by integrating modern web technologies, reducing the complexity of image management, and offering a user-friendly design. Future enhancements could include advanced search filters, image categorization, or integration with additional image APIs to expand its capabilities.

Design :

Pictopia's modular, scalable design uses the MEAN stack—MongoDB, Express.js, Angular, and Node.js—is used in Pictopia's scalable, modular design to provide a responsive, seamless user experience. While scalability manages growth through resource allocation, modularity makes updates simple. Node.js guarantees quick, concurrent processing, Angular provides a dynamic user interface, Express.js handles server tasks, and MongoDB stores flexible data. This entire JavaScript ecosystem ensures fast load times and smooth device interactions by reducing latency, improving reliability, and accommodating user demands.

Architecture Overview

- **Frontend**: Angular provides a dynamic and interactive user interface, styled with HTML, CSS, and Bootstrap for responsiveness.
- **Backend**: Node.js and Express.js handle server-side logic and API requests.
- **Database**: MongoDB stores user data and favorite images.

- **API Layer**: RESTful APIs facilitate data exchange between the client and server, with the Unsplash API providing image search capabilities.

Project Structure

```
pictopia/
├── public/
│   ├── index.html       # Home page
│   ├── login.html       # User login page
│   ├── register.html    # User registration page
│   ├── style.css        # Main stylesheet
│   ├── style2.css       # Additional styles
│   └── index.js         # Frontend JavaScript
├── server.js            # Main backend server file
├── package.json         # Node.js dependencies and scripts
├── package-lock.json    # Lock file for dependencies
└── .env                 # Environment variables
```

User Interface Design

1. **Login & Registration Page**:
    - A clean interface with input validation.
    - JWT-based authentication.
2. **Dashboard**:
    - Displays search functionality.
    - Navigation options to access favourites.
3. **Search Section**:
    - Features an input field and button for image searches, with results displayed in a responsive grid.
4. **Favourite's Section**:
    - Lists saved images with options to remove them.

Backend API Design

- **Authentication Routes**:
  - `POST /api/auth/register` – Registers a new user.
  - `POST /api/auth/login` – Logs in a user and returns a JWT token.
- **Image Search Routes**:
  - `GET /api/images/search` – Retrieves images from the Unsplash API based on user queries.
- **Favourite's Routes**:
  - `GET /api/favorites` – Fetches the user's favorite images.
  - `POST /api/favorites` – Adds an image to the favorites list.
  - `DELETE /api/favorites` – Removes an image from the favorites list.

Security Design:

- **JWT Tokens**: Used for session management and protecting routes.
- **Password Hashing**: bcrypt ensures secure storage of user passwords.
- **Input Validation**: Prevents common vulnerabilities like XSS and injection attacks.

UI/UX Considerations

- **Responsive Design**: Adapts seamlessly across devices (desktops, tablets, mobiles).
- **Intuitive Layout**: Clear navigation and visual feedback for search and favorites management.

- **Future Enhancements**: Could include image previews, advanced filters, or drag-and-drop functionality.

The design ensures **Pictopia** is user-centric, secure, and extensible, providing a robust foundation for image management.

Modules:

Pictopia is structured into distinct modules, each handling specific functionalities to ensure maintainability, scalability, allowing seamless feature additions and efficient handling of increased demand without affecting the entire system.

Authentication Module

- **Purpose**: Manages user registration, login, and session handling.
- **Features**:
    - Secure registration and login with JWT-based authentication.
    - Session validation and logout functionality.
- **API Endpoints**:
    - `POST /api/auth/register` – Creates a new user account.
    - `POST /api/auth/login` – Authenticates a user and issues a token.
    - `GET /api/auth/logout` – Clears the session token.

Image Search Module

- **Purpose**: Facilitates keyword-based image searches via the Unsplash API.
- **Features**:

- Real-time search results displayed in a grid.
- Integration with the Unsplash API for image retrieval.

➢ **API Endpoints**:

- `GET /api/images/search` – Queries the Unsplash API and returns image results.

Favorites Module

➢ **Purpose**: Allows users to save, view, and remove favorite images.

➢ **Features**:

- Add images to a personalized favorites list.
- View and delete saved images.

➢ **API Endpoints**:

- `GET /api/favorites` – Retrieves the user's saved images.
- `POST /api/favorites` – Adds an image to favorites.
- `DELETE /api/favorites` – Removes an image from favorites.

User Interface Module

➢ **Purpose**: Provides a responsive and interactive frontend experience.

➢ **Features**:

- Navigation bar for accessing dashboard, search, and favorites.
- Dynamic updates to reflect user actions (e.g., search results, favorites list).

➢ **API Usage**: Consumes data from authentication, search, and favorites APIs.

Module Interaction Flow

- **Authentication**:
    - User logs in.
    - Token generated.
    - Stored for session management.
- **Image Search**:
    - User inputs keywords.
    - Search module queries Unsplash API.
    - Results displayed.
- **Favourites Management**:
    - User saves an image.
    - Favourites module updates MongoDB.
    - Refreshed list shown.

Pictopia's modular design enhances dependability and makes it easier to incorporate new features, such as sophisticated image editing, with ease.

Architecture

Pictopia follows a three-tier architecture, ensuring modularity and scalability.

1. Presentation Layer (Frontend)

- **Technologies**:
    - Angular, HTML5, CSS3, JavaScript, Bootstrap.
- **Components**:
    - Login/Registration pages for user access.

- Dashboard with search and navigation options.
- Search and favourites sections for image interaction.

➢ **Communication**:

- Sends HTTP requests to the backend via RESTful APIs.

2. Application Layer (Backend)

➢ **Technologies**:

- Node.js, Express.js, JWT, bcrypt.

➢ **Key API Endpoints**:

- Authentication: `/api/auth/register`, `/api/auth/login`.
- Image Search: `/api/images/search`.
- Favourites: `/api/favourites (GET, POST, DELETE).

➢ **Business Logic**:

- Validates user requests and manages API interactions.
- Processes Unsplash API responses and MongoDB updates.

3. Data Layer

➢ **Technologies**:

- MongoDB, Mongoose (for schema management).

➢ **Collections**:

- Users: Stores credentials and session data (passwords hashed with bcrypt).
- Favorites: Stores user-specific image data (e.g., image URLs, IDs).

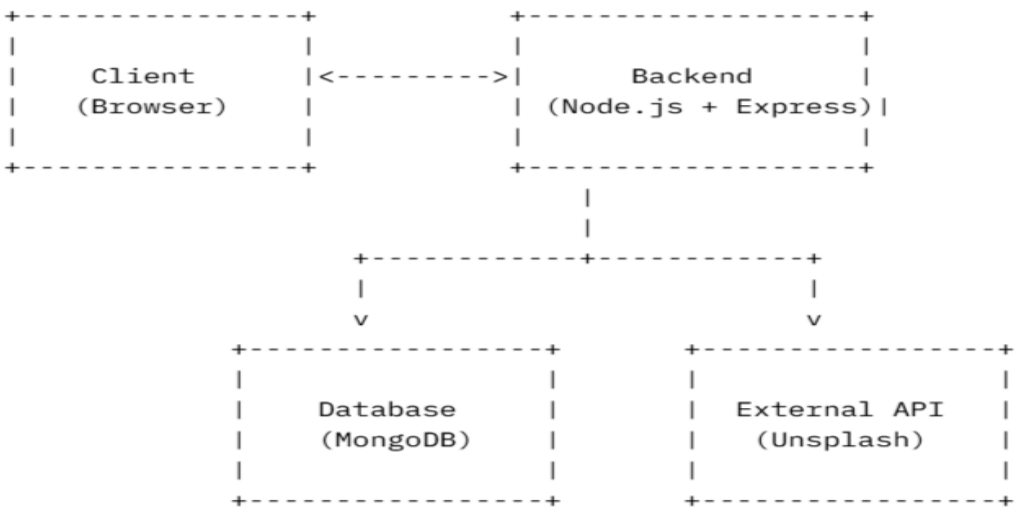
➢ **External API**:

- Unsplash API provides image search results.

Data Flow

- **Authentication**:
  - User logs in.
  - Backend validates credentials.
  - JWT issued.
  - Stored in browser.
- **Image Search**:
  - User searches.
  - Backend queries Unsplash API.
  - Results returned to frontend.
- **Favourites**:
  - User saves an image.
  - Backend updates MongoDB.
  - Frontend reflects changes.

High-Level Architecture Diagram



High-Level Architecture Text Diagram

Client (Browser)

├── Angular (HTML/CSS/JS)

└── Sends HTTP requests to Backend

|

Backend (Node.js/Express)

├── Authenticates users (JWT)

├── Queries Unsplash API

└── Interacts with MongoDB

|

Database (MongoDB)

└── Stores user data & favourites

|

External API (Unsplash)

└── Provides image search results

Security and Scalability

- **Security**: JWT for authentication, bcrypt for password hashing, and input validation.
- **Scalability**: Cloud hosting (e.g., AWS) and microservices could be adopted for growth.

This architecture ensures **Pictopia** is secure, efficient, and ready for future enhancements.

Technologies

Pictopia uses current web technologies to provide a flawless and safe experience.

1. Frontend Technologies

- **HTML5**: Structures the application's pages.
- **CSS3**: Styles the interface, with Bootstrap for responsiveness.
- **JavaScript**: Adds interactivity (via Angular).
- **Angular**: Builds a dynamic, single-page application.
- **Bootstrap**: Enhances UI with responsive design components.

2. Backend Technologies

- **Node.js**: Runs server-side JavaScript.
- **Express.js**: Manages routing and API endpoints.
- **JWT**: Secures authentication and session management.
- **bcrypt**: Hashes passwords for secure storage.
- **Mongoose**: Simplifies MongoDB interactions.

3. Database Technologies

- **MongoDB**: Stores user and favorites data in a NoSQL format.
- **Mongoose**: Provides schema validation and querying.

4. Security & Authentication

- **JWT**: Protects routes and manages sessions.
- **bcrypt**: Ensures password security.
- **CORS**: Enables safe cross-origin requests.

5. Development & Deployment Tools

- **VS Code**: Primary code editor.
- **Git/GitHub**: Version control and collaboration.
- **npm**: Package management.
- **Postman**: API testing.

6. APIs & Communication

- **RESTful APIs**: Connect frontend and backend.
- **Unsplash API**: Supplies image data.
- **Fetch API**: Handles asynchronous requests.

7. Future Enhancements

- **Docker**: For containerized deployments.
- **Cloud Platforms**: AWS/Azure for scalability.
- **CI/CD**: Automates testing and deployment.

Pictopia's MEAN stack delivers dependable performance, robust security, and an intuitive interface for users.

Coding :

The codebase for **Pictopia** is organized into frontend and backend directories, with a clear separation of concerns. The frontend, built with Angular, includes components for authentication, search, and favourites management. The backend, powered by Node.js and Express.js, handles API routes, Unsplash API integration, and MongoDB interactions. Detailed code implementation is provided in separate project files, adhering to best practices for modularity and maintainability.

Frontend: **:**

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
 <title>Pictopia</title>
 <link rel="stylesheet" href="style2.css" />

</head>
```

```
<body>
 <div class="container">
  <h1>Pictopia</h1>
  <nav>
   <a href="#" id="search-link">Search</a> |
   <a href="#" id="favorites-link">Favorites</a> |
   <span id="auth-links">
    <a href="/login">Login</a> |
    <a href="/register">Register</a>
   </span>
   <a href="#" id="logout-link">Logout</a>
  </nav>

  <div id="search-section">
   <form id="search-form">
    <input type="text" id="search-input" placeholder="Search for images..." />
    <button id="search-button" type="submit">Search</button>
   </form>

   <div id="search-results" class="search-results"></div>
   <button id="show-more-button">Show more</button>
  </div>

  <div id="favorites-section" style="display: none;">
   <h2 style="text-align:center; color:#0277bd;">Your Favorite Images</h2>
```

```
    <div id="favorites-list" class="search-results"></div>
   </div>
  </div>

  <script src="index.js"></script>
</body>
</html>
```

login.html

```
<!DOCTYPE html>

<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Login</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
  <div class="container">
    <h1>Login</h1>
    <form id="login-form" class="auth-form">
      <label for="email">Email</label>
```

```
    <input type="email" id="email" name="email" placeholder="example@gmail.com"
required />

    <label for="password">Password</label>
    <input type="password" id="password" name="password" placeholder="Enter your
password" required />

    <button type="submit">Login</button>
  </form>

  <div class="links">
    <a href="/register">Register</a> |
    <a href="/">Back to Home</a>
  </div>
 </div>

 <script>
  document.getElementById('login-form').addEventListener('submit', async (e) => {
    e.preventDefault();
    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;

    try {
      const response = await fetch('/api/auth/login', {
        method: 'POST',
```

```
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ email, password })
    });

    const data = await response.json();

    if (response.ok) {
      localStorage.setItem('token', data.token);
      window.location.href = '/';
    } else {
      alert(data.message || 'Login failed');
    }
  } catch (error) {
    alert('An error occurred. Please try again.');
  }
 });
 </script>
</body>
</html>
```

register.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Register</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
  <div class="container">
    <h1>Register</h1>
    <form id="register-form" class="auth-form">
      <label for="username">Username</label>
      <input type="text" id="username" name="username" placeholder="Enter your username"
required />

      <label for="email">Email</label>
      <input type="email" id="email" name="email" placeholder="example@gmail.com"
required />

      <label for="password">Password</label>
```

```
    <input type="password" id="password" name="password" placeholder="Create a
password" required minlength="6" />

    <button type="submit">Register</button>
  </form>

  <div class="links">
    <a href="/login">Login</a> |
    <a href="/">Back to Home</a>
  </div>
 </div>

 <script>
  document.getElementById('register-form').addEventListener('submit', async (e) => {
    e.preventDefault();
    const username = document.getElementById('username').value;
    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;

    if (password.length < 6) {
      alert('Password must be at least 6 characters long.');
      return;
    }

    try {
```

```
      const response = await fetch('/api/auth/register', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ username, email, password })
      });

      const data = await response.json();

      if (response.ok) {
        alert('Registration successful! Please login.');
        window.location.href = '/login';
      } else {
        alert(data.message || 'Registration failed');
      }
    } catch (error) {
      alert('An error occurred. Please try again.');
    }
  });
 </script>
</body>
</html>
```

style.css

```
/* General Page Styling */
body {
  margin: 0;
  padding: 0;
  background: linear-gradient(to bottom right, #e0f7fa, #b2ebf2);
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

/* Container */
.container {
  max-width: 400px;
  margin: 80px auto;
  padding: 30px;
  background-color: white;
  border-radius: 10px;
  box-shadow: 0 8px 20px rgba(0, 0, 0, 0.2);
}

/* Header */
h1 {
  text-align: center;
  color: #0277bd;
  margin-bottom: 30px;
```

```
}

/* Form Styling */
.auth-form {
  display: flex;
  flex-direction: column;
}

.auth-form label {
  margin-bottom: 5px;
  font-weight: 600;
  color: #0277bd;
}

.auth-form input {
  padding: 12px;
  font-size: 15px;
  border: 1px solid #b0bec5;
  border-radius: 5px;
  margin-bottom: 20px;
  transition: border-color 0.3s;
}

.auth-form input:focus {
  outline: none;
```

```
  border-color: #0288d1;
}

.auth-form button {
  padding: 12px;
  font-size: 16px;
  background-color: #4caf50;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.auth-form button:hover {
  background-color: #388e3c;
}

/* Links */
.links {
  text-align: center;
  margin-top: 20px;
}

.links a {
```

```
  color: #d81b60;
  text-decoration: none;
  margin: 0 10px;
}

.links a:hover {
  text-decoration: underline;
}

/* Responsive */
@media screen and (max-width: 480px) {
  .container {
    margin: 30px 20px;
    padding: 20px;
  }

  .auth-form input {
    font-size: 14px;
  }

  .auth-form button {
    font-size: 15px;
  }
}
```

Style2.css

```
/* General Styling */
body {
  margin: 0;
  padding: 0;
  background: linear-gradient(to bottom right, #e0f7fa, #b2ebf2);
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

/* Header*/

h1 {
  text-align: center;
  font-family: 'Pacifico', cursive;
  font-size: 42px;
  color: #0277bd;
  text-align: center;
  margin: 40px 0 20px;
}


/* Navigation */
```

```
nav {
  text-align: center;
  margin-bottom: 30px;
}

nav a {
  margin: 0 15px;
  color: #d81b60;
  font-weight: bold;
  text-decoration: none;
}

nav a:hover {
  text-decoration: underline;
}

/* Search Section */
#search-form {
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
  margin-bottom: 30px;
}

#search-input {
```

```
  width: 60%;
  max-width: 400px;
  padding: 12px;
  border: 1px solid #b0bec5;
  border-radius: 5px;
  font-size: 16px;
  margin-bottom: 10px;
}

#search-button {
  height: 43px;
  padding: 12px 20px;
  margin-left: 10px;
  background-color: #4caf50;
  color: white;
  border: none;
  font-size: 16px;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease-in-out;
}

#search-button:hover {
  background-color: #388e3c;
}
```

```css
/* Results Section */
.search-results {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  gap: 20px;
  padding: 20px;
  max-width: 1200px;
  margin: 0 auto;
}

.search-result {
  width: 30%;
  background: white;
  border-radius: 8px;
  box-shadow: 0 0 8px rgba(0,0,0,0.2);
  overflow: hidden;
  text-align: center;
  transition: transform 0.2s ease-in-out;
}

.search-result img {
  width: 100%;
  height: 200px;
```

```
  object-fit: cover;
  transition: transform 0.3s;
}

.search-result:hover img {
  transform: scale(1.05);
}

.search-result a {
  display: block;
  padding: 10px;
  color: #d81b60;
  text-decoration: none;
}

.search-result:hover a {
  background-color: rgba(0, 0, 0, 0.05);
}

/* Show More Button */
#show-more-button {
  display: none;
  margin: 20px auto;
  padding: 10px 20px;
  background-color: #008cba;
```

```
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

#show-more-button:hover {
  background-color: #1c506c;
}

/* Favorites Section */
.favorite-btn {
  background: none;
  border: none;
  font-size: 24px;
  cursor: pointer;
  color: red;
  transition: color 0.3s ease;
}

.favorite-btn.favorited {
  color: blue;
}
```

```
.remove-favorite {
  background: #f44336;
  color: white;
  border: none;
  padding: 8px 12px;
  margin-top: 10px;
  cursor: pointer;
  border-radius: 4px;
}



/* Responsive */
@media screen and (max-width: 768px) {
  .search-result {
    width: 45%;
  }

  #search-input {
    width: 80%;
  }
}

@media screen and (max-width: 480px) {
  .search-result {
    width: 100%;
```

```
  }

  #search-form {
    flex-direction: column;
    align-items: center;
  }

  #search-button {
    margin-left: 0;
  }
}
```

Index.js

```
const form = document.getElementById('search-form');

const searchInput = document.getElementById('search-input');
const searchResults = document.getElementById('search-results');
const showMoreButton = document.getElementById('show-more-button');
const favoritesList = document.getElementById('favorites-list');
const searchSection = document.getElementById('search-section');
const favoritesSection = document.getElementById('favorites-section');
const searchLink = document.getElementById('search-link');
const favoritesLink = document.getElementById('favorites-link');
const authLinks = document.getElementById('auth-links');
const logoutLink = document.getElementById('logout-link');
```

```
let query = '';
let page = 1;

const token = localStorage.getItem('token');
if (token) {
  authLinks.style.display = 'none';
  logoutLink.style.display = 'inline';
} else {
  authLinks.style.display = 'inline';
  logoutLink.style.display = 'none';
}

logoutLink.addEventListener('click', (e) => {
  e.preventDefault();
  localStorage.removeItem('token');
  window.location.href = '/login';
});

searchLink.addEventListener('click', (e) => {
  e.preventDefault();
  searchSection.style.display = 'block';
  favoritesSection.style.display = 'none';
});
```

```
favoritesLink.addEventListener('click', (e) => {
  e.preventDefault();
  searchSection.style.display = 'none';
  favoritesSection.style.display = 'block';
  displayFavorites();
});

form.addEventListener('submit', async (e) => {
  e.preventDefault();
  if (!token) {
    alert('Please login to search images.');
    window.location.href = '/login';
    return;
  }
  query = searchInput.value.trim();
  if (!query) {
    alert('Please enter a search query.');
    return;
  }
  page = 1;
  searchImages();
});

showMoreButton.addEventListener('click', () => {
  page++;
```

```
  searchImages();
});

async function searchImages() {
  try {
    const response = await fetch(`/api/images/search?page=${page}&query=${query}`, {
      headers: { 'Authorization': `Bearer ${token}` }
    });
    const data = await response.json();

    if (page === 1) searchResults.innerHTML = '';

    if (response.ok) {
      const favorites = await getFavorites();
      data.results.forEach(photo => {
        const isFavorited = favorites.some(img => img.id === photo.id);
        const result = document.createElement('div');
        result.classList.add('search-result');
        result.innerHTML = `
          <img src="${photo.urls.small}" alt="${photo.alt_description || 'Image'}" />
          <a href="${photo.links.html}" target="_blank">${photo.alt_description || 'View
Image'}</a>
          <button class="favorite-btn ${isFavorited ? 'favorited' : ''}"
                  data-image-id="${photo.id}"
                  data-url="${photo.urls.small}"
```

```
            data-desc="${photo.alt_description || 'Image'}">♡</button>
        `;
        searchResults.appendChild(result);
      });
      showMoreButton.style.display = data.results.length > 0 ? 'block' : 'none';
    } else {
      alert(data.message || 'Failed to load images');
    }
  } catch (error) {
    console.error('Error:', error);
    alert('Failed to load images. Please try again.');
  }
}

searchResults.addEventListener('click', async (e) => {
  if (e.target.classList.contains('favorite-btn')) {
    const button = e.target;
    const imageId = button.getAttribute('data-image-id');
    const imageUrl = button.getAttribute('data-url');
    const imageDesc = button.getAttribute('data-desc');
    await toggleFavorite(imageId, imageUrl, imageDesc, button);
  }
});

async function toggleFavorite(id, url, desc, button) {
```

```
  try {
    const method = button.classList.contains('favorited') ? 'DELETE' : 'POST';
    const response = await fetch('/api/favorites', {
      method,
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${token}`
      },
      body: JSON.stringify({ id, url, desc })
    });

    if (response.ok) {
      button.classList.toggle('favorited');
      button.innerHTML = button.classList.contains('favorited') ? '💚' : '💚';
    } else {
      const data = await response.json();
      alert(data.message || 'Failed to update favorite');
    }
  } catch (error) {
    alert('Error updating favorite');
  }
}

async function displayFavorites() {
  favoritesList.innerHTML = '';
```

```
 try {
  const favorites = await getFavorites();
  if (favorites.length === 0) {
   favoritesList.innerHTML = '<p>No favorite images yet.</p>';
  } else {
   favorites.forEach(img => {
    const div = document.createElement('div');
    div.classList.add('search-result');
    div.innerHTML = `
     <img src="${img.url}" alt="${img.desc}" />
     <p>${img.desc}</p>
     <button class="remove-favorite" data-image-id="${img.id}">Remove</button>
    `;
    favoritesList.appendChild(div);
   });
  }
 } catch (error) {
  favoritesList.innerHTML = '<p>Error loading favorites</p>';
 }
}

async function getFavorites() {
 const response = await fetch('/api/favorites', {
  headers: { 'Authorization': `Bearer ${token}` }
 });
```

```
  return response.ok ? await response.json() : [];
}


favoritesList.addEventListener('click', async (e) => {
  if (e.target.classList.contains('remove-favorite')) {
    const imageId = e.target.getAttribute('data-image-id');
    await fetch('/api/favorites', {
      method: 'DELETE',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${token}`
      },
      body: JSON.stringify({ id: imageId })
    });
    displayFavorites();
    updateFavoriteButtons();
  }
});


function updateFavoriteButtons() {
  getFavorites().then(favorites => {
    const favoriteBtns = searchResults.querySelectorAll('.favorite-btn');
    favoriteBtns.forEach(btn => {
      const imageId = btn.getAttribute('data-image-id');
      btn.classList.toggle('favorited', favorites.some(img => img.id === imageId));
```

```
  });
 });
}
```

`Backend

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const jwt = require('jsonwebtoken');
const bcrypt = require('bcryptjs');
const axios = require('axios');
require('dotenv').config();

const app = express();
app.use(express.json());
app.use(express.static('public')); // Serve static files

// MongoDB Connection
mongoose.connect(process.env.MONGODB_URI)
 .then(() => console.log('MongoDB connected'))
 .catch(err => console.error('MongoDB connection error:', err));

// User Schema
```

```
const userSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  favorites: [{ id: String, url: String, desc: String }]
});

const User = mongoose.model('User', userSchema);

// Middleware to verify JWT
const authenticateToken = (req, res, next) => {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];
  if (!token) return res.status(401).json({ message: 'Authentication required' });

  jwt.verify(token, process.env.JWT_SECRET, (err, user) => {
    if (err) return res.status(403).json({ message: 'Invalid token' });
    req.user = user;
    next();
  });
};

// Routes
app.post('/api/auth/register', async (req, res) => {
  try {
```

```
    const { username, email, password } = req.body;
    const hashedPassword = await bcrypt.hash(password, 10);
    const user = new User({ username, email, password: hashedPassword });
    await user.save();
    res.status(201).json({ message: 'User registered successfully' });
  } catch (error) {
    res.status(400).json({ message: error.message });
  }
});

app.post('/api/auth/login', async (req, res) => {
  try {
    const { email, password } = req.body;
    const user = await User.findOne({ email });
    if (!user || !(await bcrypt.compare(password, user.password))) {
      return res.status(401).json({ message: 'Invalid credentials' });
    }
    const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, { expiresIn: '1h' });
    res.json({ token });
  } catch (error) {
    res.status(500).json({ message: 'Server error' });
  }
});

app.get('/api/images/search', authenticateToken, async (req, res) => {
```

```
  try {
    const { page, query } = req.query;
    const response = await axios.get(`https://api.unsplash.com/search/photos`, {
      params: { page, query, client_id: process.env.UNSPLASH_ACCESS_KEY }
    });
    res.json(response.data);
  } catch (error) {
    res.status(500).json({ message: 'Error fetching images' });
  }
});

app.get('/api/favorites', authenticateToken, async (req, res) => {
  try {
    const user = await User.findById(req.user.id);
    res.json(user.favorites);
  } catch (error) {
    res.status(500).json({ message: 'Error fetching favorites' });
  }
});

app.post('/api/favorites', authenticateToken, async (req, res) => {
  try {
    const { id, url, desc } = req.body;
    const user = await User.findById(req.user.id);
    if (!user.favorites.some(fav => fav.id === id)) {
```

```
      user.favorites.push({ id, url, desc });
      await user.save();
    }
    res.status(200).json({ message: 'Added to favorites' });
  } catch (error) {
    res.status(500).json({ message: 'Error adding favorite' });
  }
});

app.delete('/api/favorites', authenticateToken, async (req, res) => {
  try {
    const { id } = req.body;
    const user = await User.findById(req.user.id);
    user.favorites = user.favorites.filter(fav => fav.id !== id);
    await user.save();
    res.status(200).json({ message: 'Removed from favorites' });
  } catch (error) {
    res.status(500).json({ message: 'Error removing favorite' });
  }
});

// Serve HTML pages
app.get('/', (req, res) => res.sendFile(__dirname + '/public/index.html'));
app.get('/login', (req, res) => res.sendFile(__dirname + '/public/login.html'));
app.get('/register', (req, res) => res.sendFile(__dirname + '/public/register.html'));
```

```
const PORT = process.env.PORT || 3001;

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

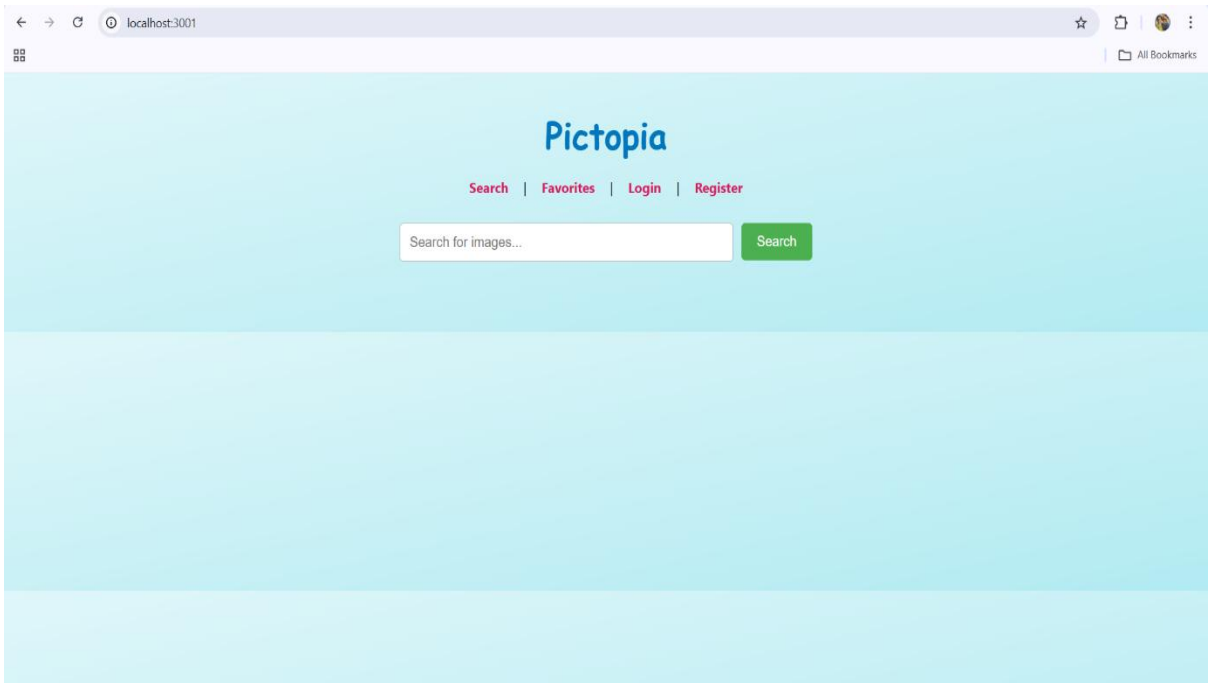
package.json

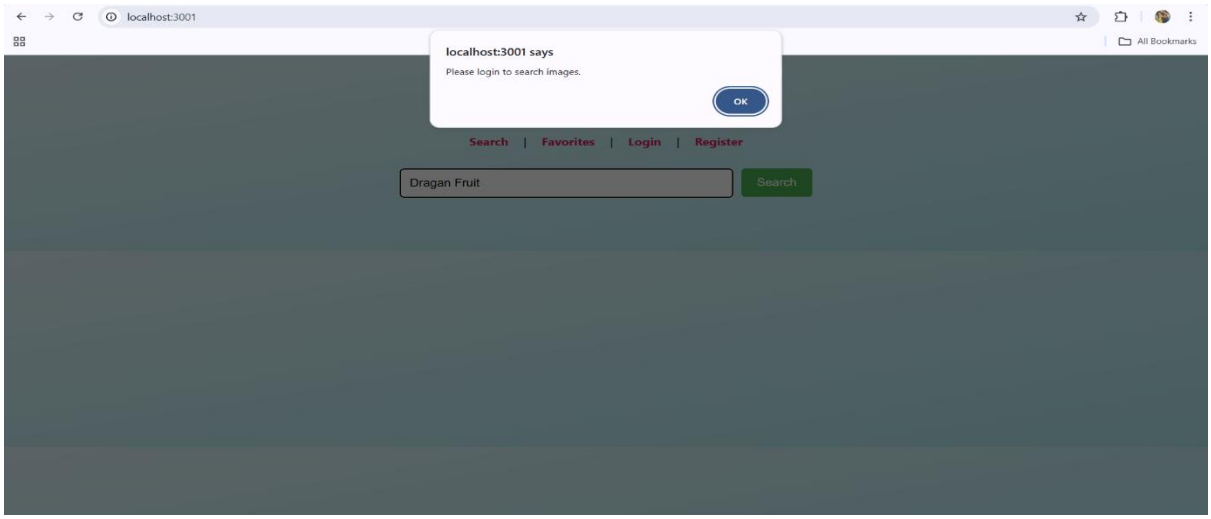
```
{
 "name": "pictopia",
 "version": "1.0.0",
 "main": "server.js",
 "scripts": {
   "start": "node server.js"
 },
 "dependencies": {
   "axios": "^1.6.0",
   "bcryptjs": "^2.4.3",
   "dot": "^1.1.3",
   "dotenv": "^16.4.7",
   "env": "^0.0.2",
   "express": "^4.18.2",
   "jsonwebtoken": "^9.0.0",
   "mongoose": "^7.0.0"
 }
}
```

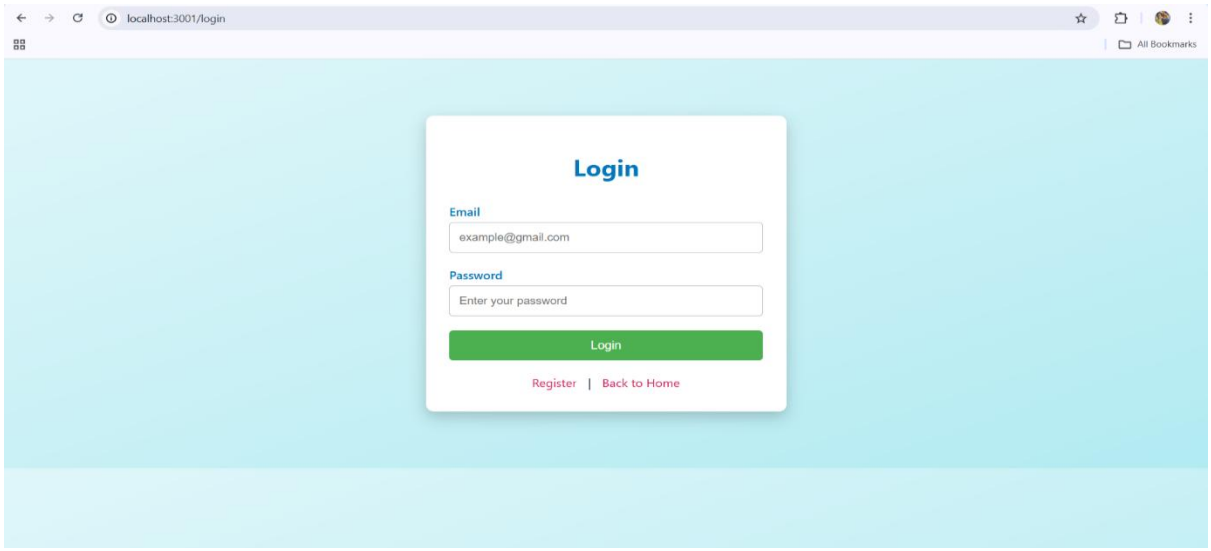
.env

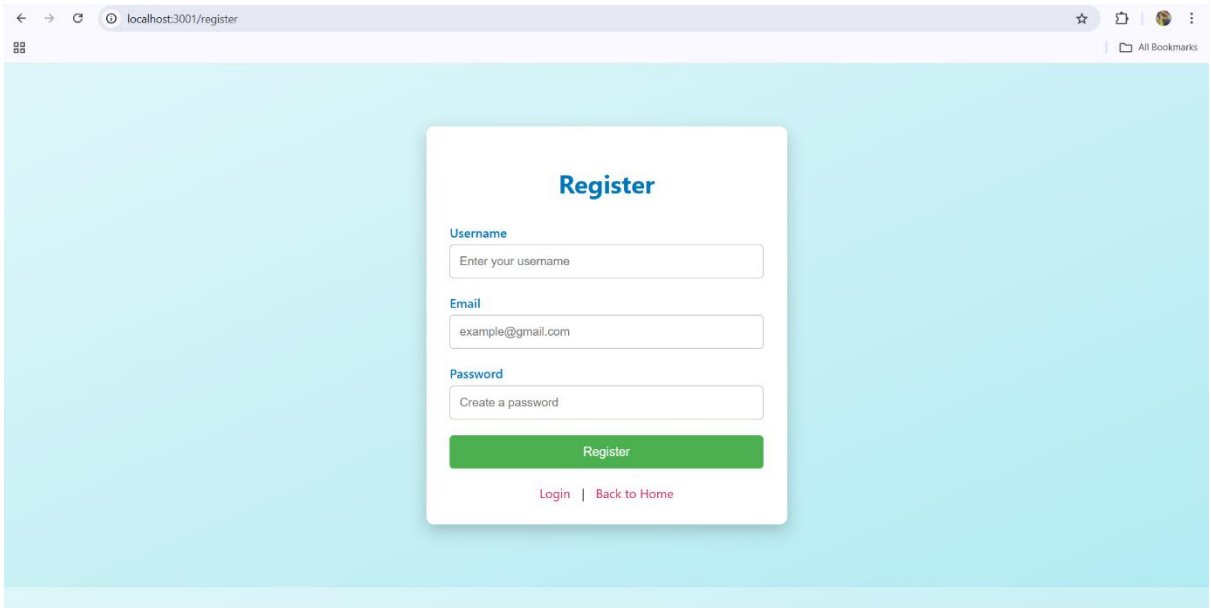
```
MONGODB_URI=mongodb+srv://Pictopia:Pictopia2921@pictopia.lrw6yt8.mongodb.net/?retryWrites=true&w=majority&appName=Pictopia
JWT_SECRET=Your-Secret-Key-iS-HeRE-usHhhh
UNSPLASH_ACCESS_KEY=MzLBGMdo0MopaxeMaH0wgaQ4EKP_kiNm76b1jxnU2IQ
PORT=3001
```

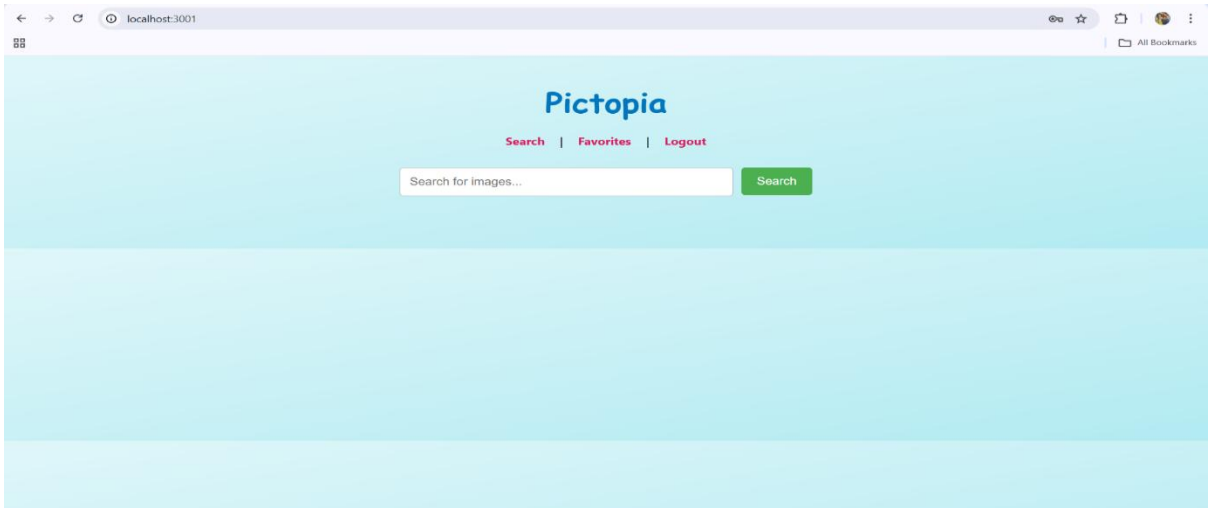
Screenshots

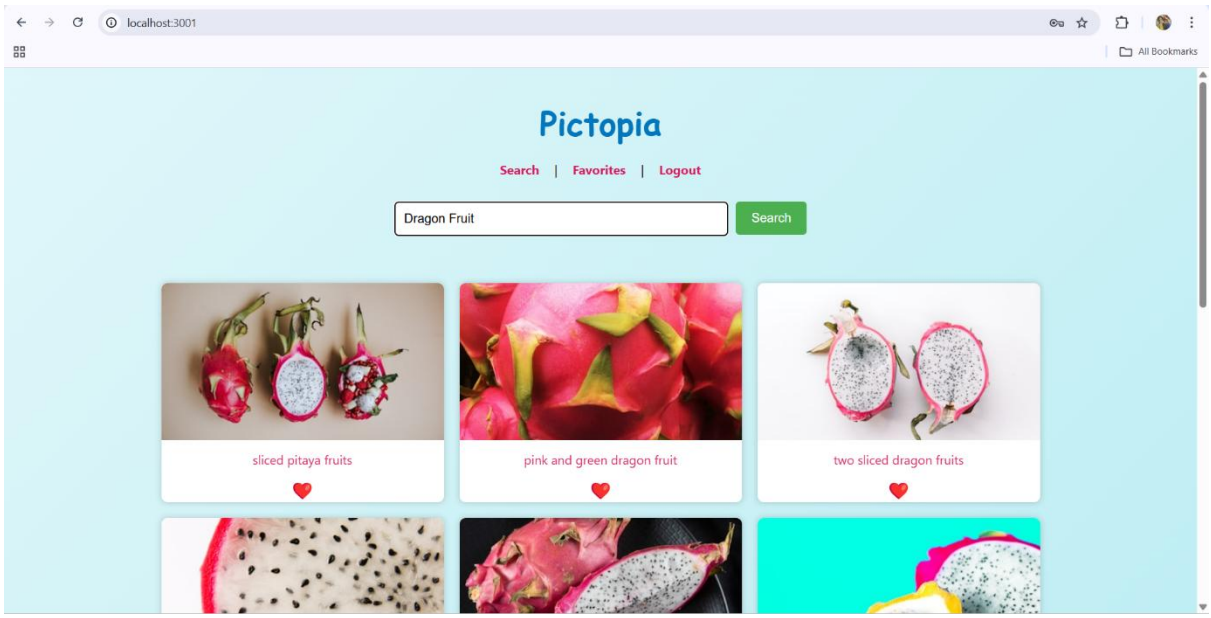


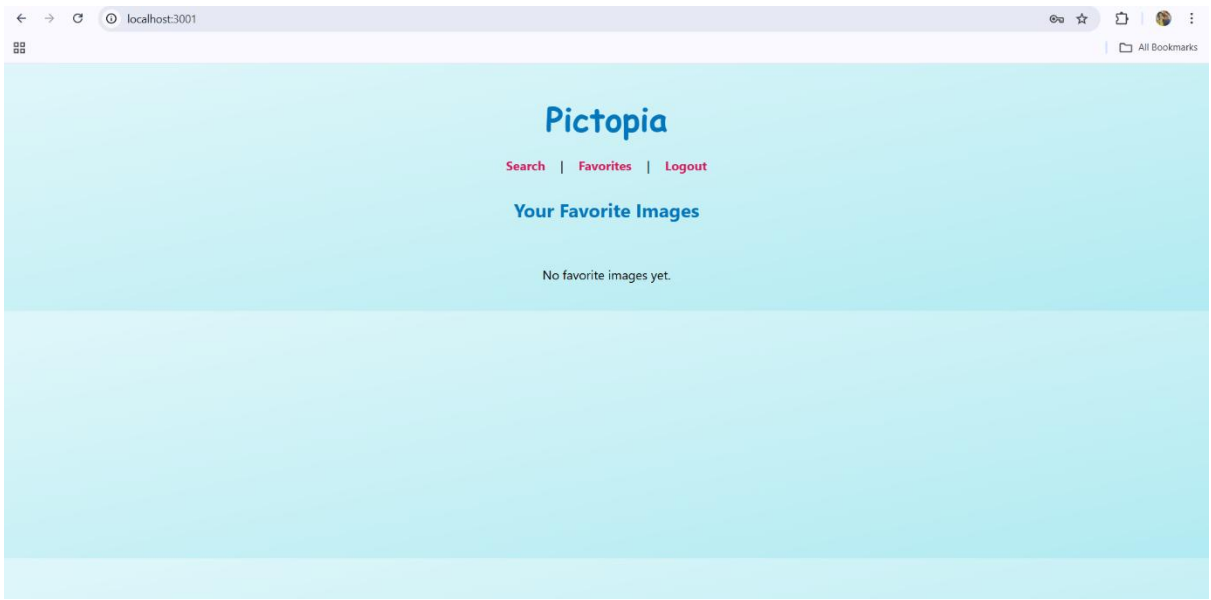

localhost:3001
All Bookmarks
localhost:3001 says
Please login to search images.
OK
Search | Favorites | Login | Register
Dragan Fruit
Search

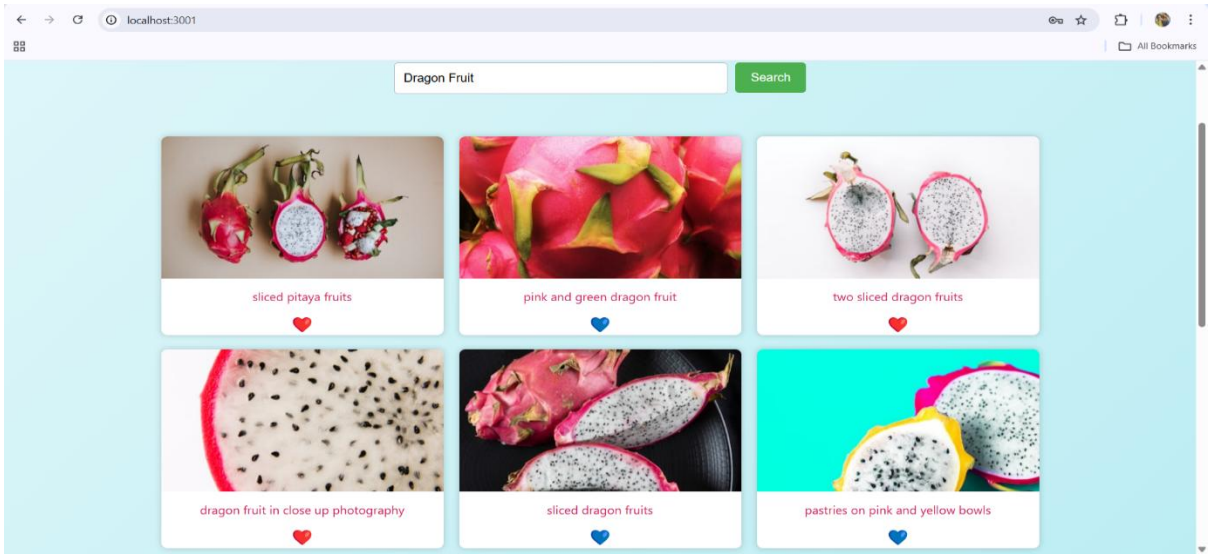

localhost:3001/login
All Bookmarks
Login
Email
example@gmail.com
Password
Enter your password
Login
Register | Back to Home

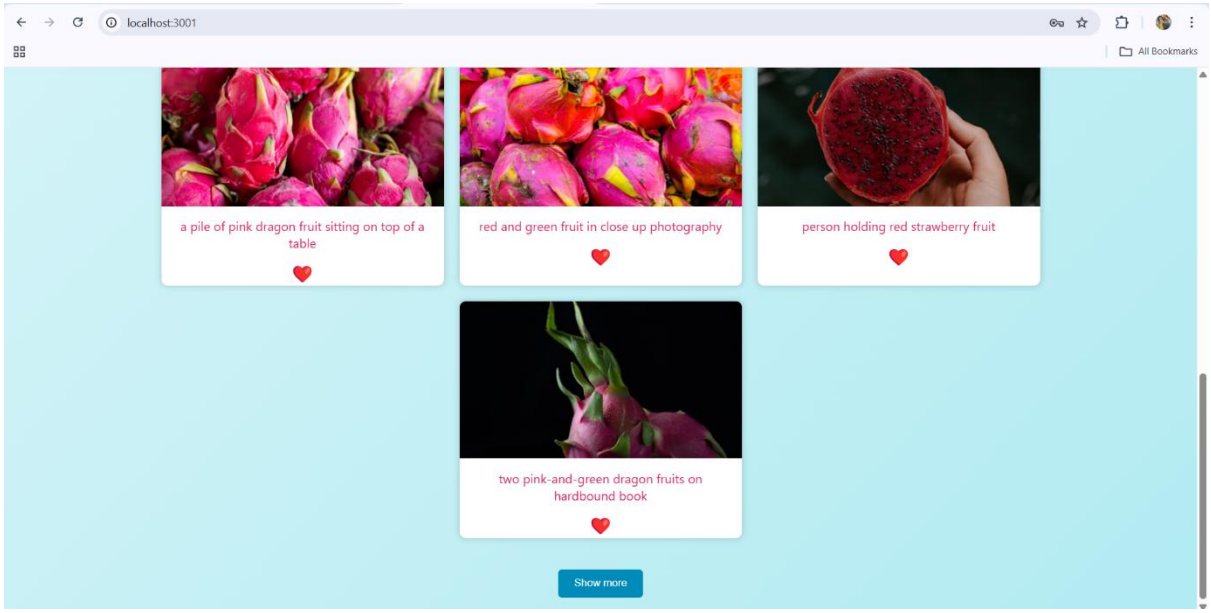

localhost:3001/register
All Bookmarks
Register
Username
Enter your username
Email
example@gmail.com
Password
Create a password
Register
Login | Back to Home

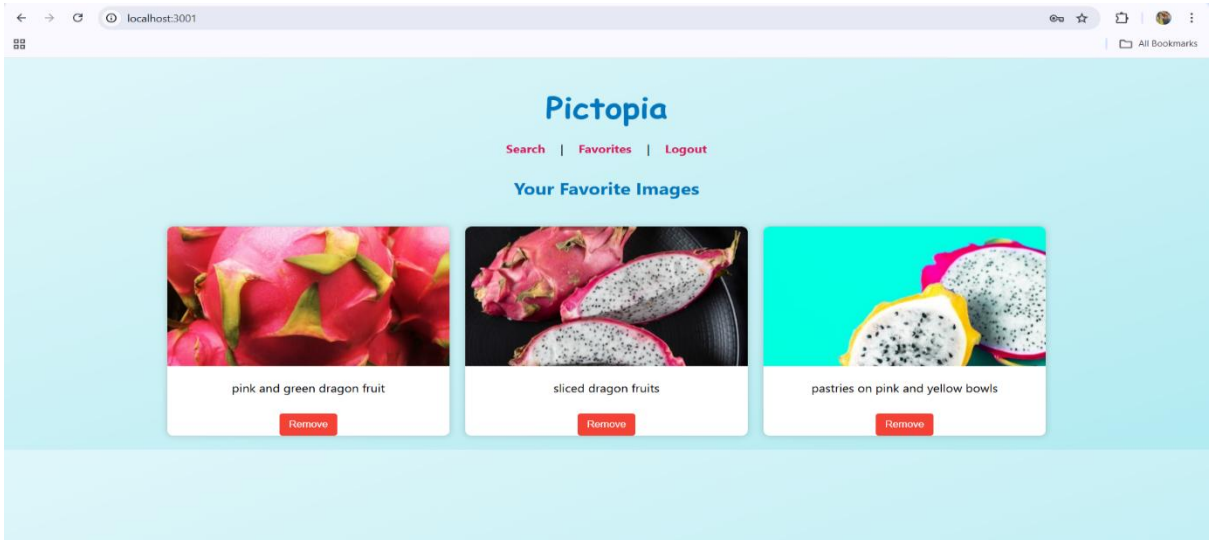

localhost:3001
All Bookmarks
Pictopia
Search | Favorites | Logout
Search for images...
Search

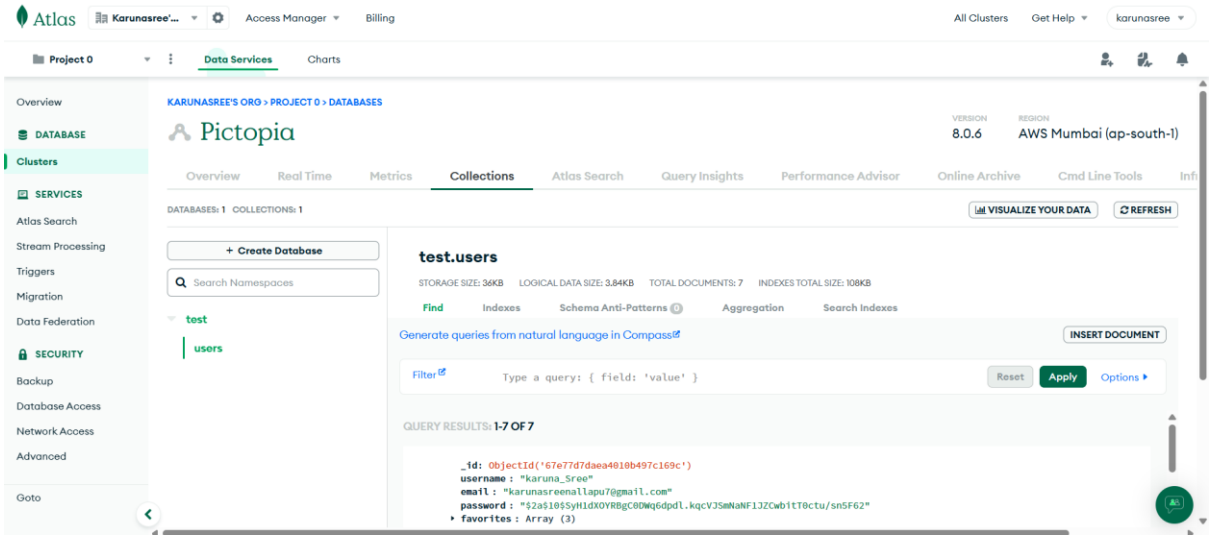

localhost:3001
All Bookmarks
Pictopia
Search | Favorites | Logout
Dragon Fruit
Search
sliced pitaya fruits
pink and green dragon fruit
two sliced dragon fruits

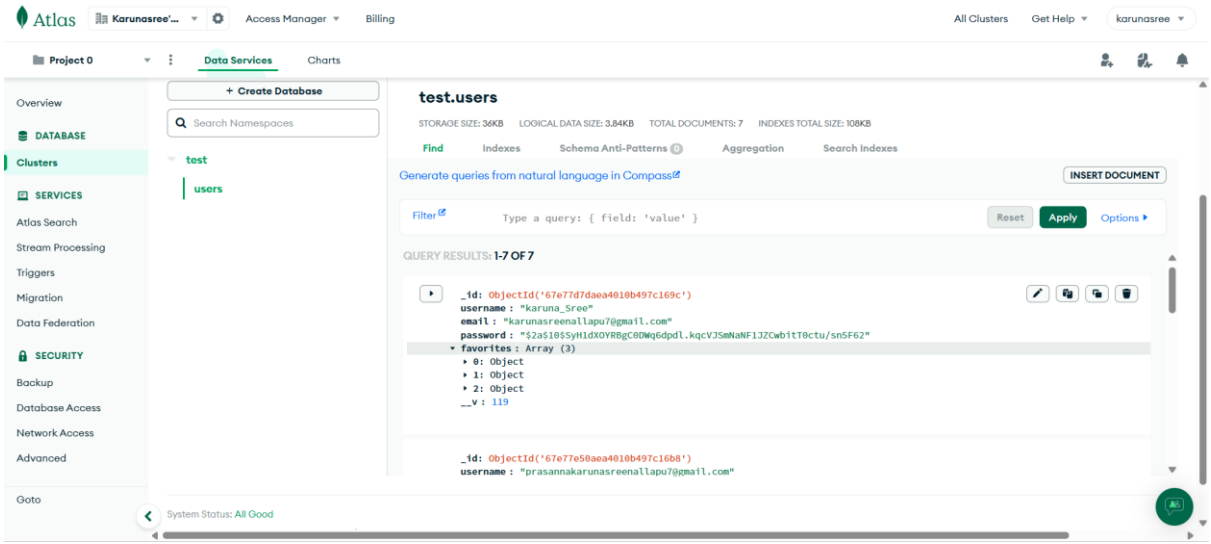

localhost:3001
All Bookmarks
Pictopia
Search | Favorites | Logout
Your Favorite Images
No favorite images yet.

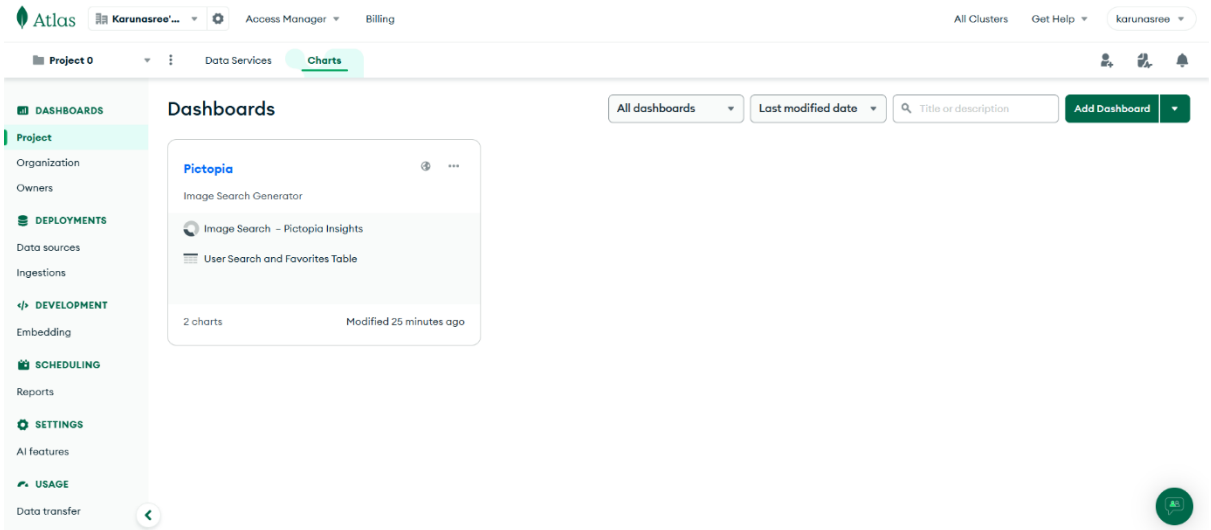

localhost:3001
All Bookmarks
Dragon Fruit
Search
sliced pitaya fruits
pink and green dragon fruit
two sliced dragon fruits
dragon fruit in close up photography
sliced dragon fruits
pastries on pink and yellow bowls

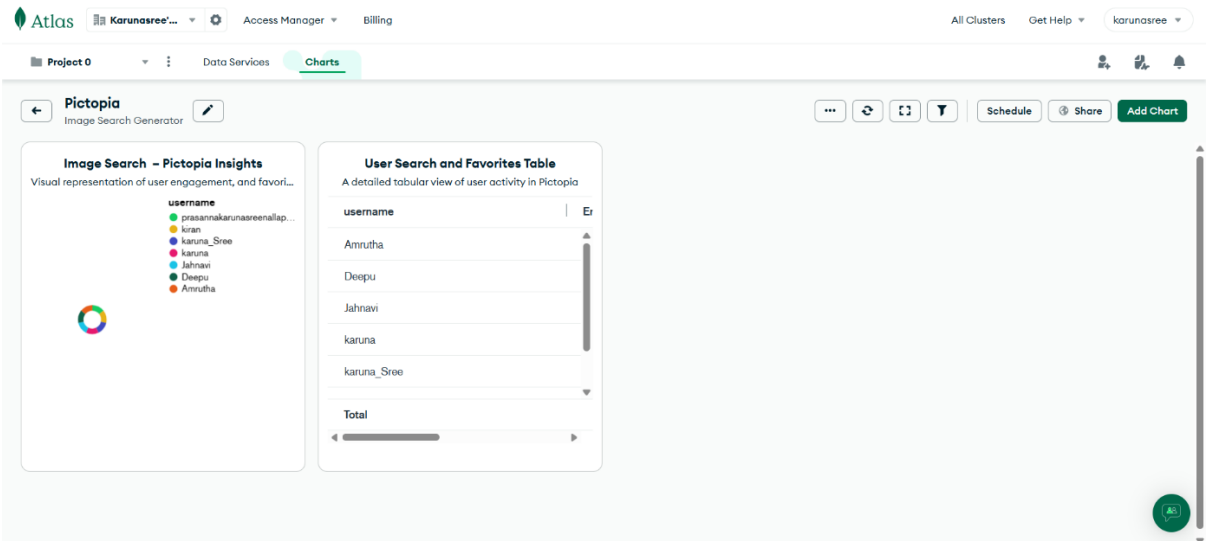

localhost:3001
All Bookmarks
a pile of pink dragon fruit sitting on top of a table
red and green fruit in close up photography
person holding red strawberry fruit
two pink-and-green dragon fruits on hardbound book
Show more


localhost:3001
All Bookmarks
Pictopia
Search | Favorites | Logout
Your Favorite Images
pink and green dragon fruit
Remove
sliced dragon fruits
Remove
pastries on pink and yellow bowls
Remove


Atlas
Karunasree'...
Access Manager
Billing
All Clusters
Get Help
karunasree
Project 0
Data Services
Charts
Overview
DATABASE
Clusters
SERVICES
Atlas Search
Stream Processing
Triggers
Migration
Data Federation
SECURITY
Backup
Database Access
Network Access
Advanced
Goto
KARUNASREE'S ORG > PROJECT 0 > DATABASES
Pictopia
VERSION
8.0.6
REGION
AWS Mumbai (ap-south-1)
Overview
Real Time
Metrics
Collections
Atlas Search
Query Insights
Performance Advisor
Online Archive
Cmd Line Tools
DATABASES: 1 COLLECTIONS: 1
VISUALIZE YOUR DATA
REFRESH
+ Create Database
Search Namespaces
test
users
test.users
STORAGE SIZE: 36KB
LOGICAL DATA SIZE: 3.84KB
TOTAL DOCUMENTS: 7
INDEXES TOTAL SIZE: 108KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
Generate queries from natural language in Compass
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
QUERY RESULTS: 1-7 OF 7
_id: ObjectId('67e77d7daea4010b497c169c')
username : "karuna_Sree"
email : "karunasreenallapu7@gmail.com"
password : "$2a$10$SyH1dXOYRBgC0DWq6dpdl.kqcVJSmNaNF1JZCwbitT0ctu/sn5F62"
favorites : Array (3)


Atlas
Karunasree'...
Access Manager
Billing
All Clusters
Get Help
karunasree
Project 0
Data Services
Charts
Overview
DATABASE
Clusters
SERVICES
Atlas Search
Stream Processing
Triggers
Migration
Data Federation
SECURITY
Backup
Database Access
Network Access
Advanced
Goto
+ Create Database
Search Namespaces
test
users
test.users
STORAGE SIZE: 36KB
LOGICAL DATA SIZE: 3.84KB
TOTAL DOCUMENTS: 7
INDEXES TOTAL SIZE: 108KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
Generate queries from natural language in Compass
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
QUERY RESULTS: 1-7 OF 7
_id: ObjectId('67e77d7daea4010b497c169c')
username : "karuna_Sree"
email : "karunasreenallapu7@gmail.com"
password : "$2a$10$SyH1dXOYRBgC0DWq6dpdl.kqcVJSmNaNF1JZCwbitT0ctu/sn5F62"
favorites : Array (3)
0: Object
1: Object
2: Object
__v : 119
_id: ObjectId('67e77e50aea4010b497c16b8')
username : "prasannakarunasreenallapu7@gmail.com"
System Status: All Good


Atlas
Karunasree'...
Access Manager
Billing
All Clusters
Get Help
karunasree
Project 0
Data Services
Charts
DASHBOARDS
Project
Organization
Owners
DEPLOYMENTS
Data sources
Ingestions
DEVELOPMENT
Embedding
SCHEDULING
Reports
SETTINGS
AI features
USAGE
Data transfer
Dashboards
All dashboards
Last modified date
Title or description
Add Dashboard
Pictopia
Image Search Generator
Image Search – Pictopia Insights
User Search and Favorites Table
2 charts
Modified 25 minutes ago


Atlas
Karunasree'...
Access Manager
Billing
All Clusters
Get Help
karunasree
Project 0
Data Services
Charts
Pictopia
Image Search Generator
Schedule
Share
Add Chart
Image Search – Pictopia Insights
Visual representation of user engagement, and favori...
username
prasannakarunasreenallap...
kiran
karuna_Sree
karuna
Jahnavi
Deepu
Amrutha
User Search and Favorites Table
A detailed tabular view of user activity in Pictopia
username
Amrutha
Deepu
Jahnavi
karuna
karuna_Sree
Total

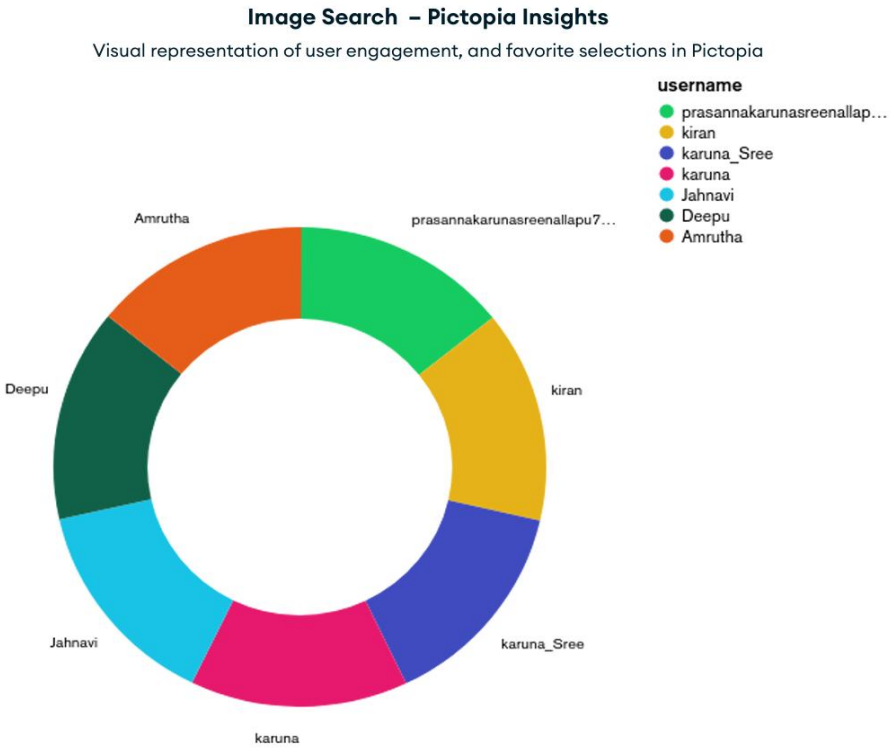


Image Search – Pictopia Insights
Visual representation of user engagement, and favorite selections in Pictopia
username
prasannakarunasreenallap...
kiran
karuna_Sree
karuna
Jahnavi
Deepu
Amrutha
Amrutha
prasannakarunasreenallapu7...
kiran
karuna_Sree
karuna
Jahnavi
Deepu

User Search and Favorites Table

A detailed tabular view of user activity in Pictopia

username	Email	Favorites
Amrutha	Other values	3
Deepu	Other values	0
Jahnavi	Other values	0
karuna	Other values	1
karuna_Sree	Other values	7
kiran	Other values	2
prasannakarunasreenallapu7@gmail.com	Other values	0
Total		**13**