You're right — the document cut off at section **14. Running the App**. Let me continue and finalize it here:

---

14. Running the App

```
# Start your server
node app.js
```

Now open http://localhost:3000 in your browser.

- If you login with a **Superadmin**, you'll be able to see all statistics and admin pages.
- If you login with an **Admin**, you'll be able to manage users but not see superadmin-only content.
- If you login with a **User**, you'll only be able to access user dashboard pages.

---

15. Folder Structure Recap (MVC)

```
project/
│   app.js
│   package.json
│
├──config/
│       db.js
│
├──models/
│       userModel.js
│
├──controllers/
│       authController.js
│       dashboardController.js
│
├──routes/
│       authRoutes.js
│       dashboardRoutes.js
│
├──middleware/
│       authMiddleware.js
│
├──public/
│   ├── css/
│   │     styles.css
│   ├── js/
│   │     scripts.js
│   └── html/
```

```
│         login.html
│         dashboard.html
│         admin.html
│         superadmin.html
```

---

16. How Frontend Checks Roles

You don't expose the session directly to the browser. Instead:

1. The backend (Node.js) checks the role using middleware.
2. Only if the user is authorized, you serve the requested page (HTML).
3. If the user tries to visit a restricted page directly (like `/dashboard/superadmin`), the middleware denies access and redirects them.

---

17. Next Steps

✅Add **database integration** (MySQL/Postgres/Mongo) instead of in-memory `users` array. ✅Secure session storage (use Redis or database store in production). ✅Add CSRF protection & input sanitization. ✅ Replace plain HTML with templating (EJS, Handlebars) or even React later.

✅Now you have a full **MVC Node.js + Sessions + Role-based Auth** system with your vanilla HTML/CSS/ Bootstrap frontend.

Do you want me to extend this with a **MySQL integration example** (instead of the in-memory array) so you can store users and roles persistently?

---

14) Frontend (public/app.html) — full file

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>App</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/
bootstrap.min.css" rel="stylesheet" />
</head>
<body>
  <nav class="navbar navbar-expand-lg bg-body-tertiary">
    <div class="container-fluid">
```

```
      <a class="navbar-brand" href="#">My App (MVC)</a>
      <div class="d-flex gap-2 ms-auto">
        <span id="whoami" class="navbar-text"></span>
        <button id="logoutBtn" class="btn btn-outline-secondary btn-sm">Logout</
button>
      </div>
    </div>
  </nav>

  <main class="container py-4">
    <div class="alert alert-info d-none" id="notAllowed">You do not have access
to this area.</div>

    <!-- Superadmin-only -->
    <section id="statistics" class="d-none">
      <h2 class="h5">Statistics (Superadmin)</h2>
      <div class="card">
        <div class="card-body">
          <p class="mb-1">Users: <strong id="usersCount">-</strong></p>
          <small class="text-muted" id="generatedAt"></small>
        </div>
      </div>
    </section>

    <!-- Staff-only example -->
    <section id="staffTools" class="d-none mt-4">
      <h2 class="h5">Staff Tools</h2>
      <div class="card"><div class="card-body">Basic staff utilities…</div></
div>
    </section>
  </main>

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

---

15) Frontend Logic (public/js/app.js) — full file

```
async function fetchMe() {
  const res = await fetch('/auth/me');
  if (!res.ok) throw new Error('Not logged in');
  return res.json();
}
```

```
function loadScript(src) {
  const s = document.createElement('script');
  s.src = src;
  document.body.appendChild(s);
}

async function init() {
  try {
    const user = await fetchMe();
    document.getElementById('whoami').textContent = `${user.email} ($
{user.role})`;

    if (user.role === 'superadmin') {
      document.getElementById('statistics').classList.remove('d-none');
      loadScript('/js/superadmin.js');
    } else if (user.role === 'staff') {
      document.getElementById('staffTools').classList.remove('d-none');
      loadScript('/js/staff.js');
    }
  } catch {
    window.location.href = '/login.html';
  }
}

const logoutBtn = document.getElementById('logoutBtn');
if (logoutBtn) {
  logoutBtn.addEventListener('click', async () => {
    await fetch('/auth/logout', { method: 'POST' });
    window.location.href = '/login.html';
  });
}

async function loadStatistics() {
  try {
    const res = await fetch('/admin/statistics');
    if (!res.ok) throw new Error('Forbidden');
    const data = await res.json();
    document.getElementById('usersCount').textContent = data.users;
    document.getElementById('generatedAt').textContent = new
Date(data.generatedAt).toLocaleString();
  } catch (e) {
    document.getElementById('notAllowed').classList.remove('d-none');
  }
}

window.loadStatistics = loadStatistics;
init();
```

16) Role-Specific Scripts

public/js/superadmin.js

```
console.log('Superadmin script loaded');
if (window.loadStatistics) window.loadStatistics();
```

public/js/staff.js

```
console.log('Staff script loaded');
```

---

17) Run the App

```
# 1) Generate a bcrypt hash and seed your user in MySQL
node scripts/hash-password.js Password123!
# 2) Start the server
node server.js
# 3) Visit
http://localhost:3000/login.html
```

---

18) Security Notes

- Cookies: `httpOnly`, `sameSite: 'lax'`, `secure: true` in production.
- Always enforce roles on the backend (`requireRole`).
- Consider CSRF protection (`csurf`) if you use many forms.
- Keep session data minimal: `{ id, email, role }` only.