EM Interview Prep - Critical Issues Fixed

Summary of Fixes Applied

Phase 1: Authentication Issues (FIXED)

Problem: Server components were calling `getSession()` without required `NextRequest` parameter, causing 307 redirects to login for protected routes.

Solution:
- Updated all protected pages to use `getServerSession()` instead of `getSession()`
- Fixed pages: Interview Strategy, Progress Tracker, FAQ, System Design Strategy, System Design Questions
- All pages now properly authenticate users and redirect to login only when necessary

Files Modified:
- `/app/interview-strategy/page.tsx`
- `/app/progress-tracker/page.tsx`
- `/app/faq/page.tsx`
- `/app/system-design-strategy/page.tsx`
- `/app/system-design-questions/page.tsx`

Phase 2: System Design Issues (FIXED)

Problem: "frameworks.map is not a function" error on SD Strategy page due to API response format mismatch.

Solution:
- Fixed API route `/api/system-design-frameworks/route.ts` to return frameworks array directly instead of wrapped object
- Ensured client component expects correct data format
- System design questions are now visible and searchable (22 questions available)

Files Modified:
- `/app/api/system-design-frameworks/route.ts`
- `/components/system-design/system-design-strategy-client.tsx`

Phase 3: UI/UX Navigation Issues (FIXED)

Problem: Dashboard horizontal tab overflow causing poor UX on smaller screens.

Solution:
- Improved header responsive design with proper flex layouts
- Added horizontal scrolling with `scrollbar-hide` for navigation overflow
- Optimized spacing and sizing for better mobile experience
- Made navigation items more compact with better text sizing

Files Modified:
- `/components/layout/header.tsx`

Phase 4: Comprehensive Testing Framework (IMPLEMENTED)

Problem: No testing framework to prevent regressions and ensure code quality.

Solution:
- Implemented Jest + React Testing Library for unit/component testing
- Created comprehensive test suites covering:
- Authentication functions and flows
- Component rendering and interactions
- API route functionality
- Integration testing for critical user workflows
- Added test scripts and configuration

Files Created:
- `jest.config.js` - Jest configuration
- `jest.setup.js` - Test environment setup
- `__tests__/auth.test.ts` - Authentication unit tests
- `__tests__/components/header.test.tsx` - Header component tests
- `__tests__/api/auth.test.ts` - API authentication tests
- `__tests__/integration/authentication.test.ts` - Integration tests
- `__tests__/system-design/frameworks.test.ts` - System design component tests

System Design Categories (Limited to 5 as requested)

The system now has the following categories for system design questions:
1. **Data Consistency** - CAP Theorem, ACID, BASE principles
2. **Architecture Patterns** - Microservices, distributed systems
3. **Scalability** - Load balancing, horizontal/vertical scaling
4. **Performance** - Caching strategies, optimization
5. **Security** - Authentication, authorization, data protection

Current System Status

Working Features

- **Authentication**: All protected routes properly authenticate users
- **System Design Questions**: 22 questions available and searchable
- **System Design Frameworks**: 6 frameworks available with filtering
- **Navigation**: Responsive header with improved mobile experience
- **Database**: All data properly seeded and accessible

Test Coverage

- Authentication functions (token verification, creation)
- Component rendering (Header, System Design components)
- API endpoints (login, frameworks, questions)
- Integration workflows (authentication flow, data fetching)

Performance Improvements

- Fixed horizontal navigation overflow
- Improved responsive design
- Better mobile experience
- Optimized component loading

Next Steps for Further Enhancement

Recommended Improvements

1. **Add End-to-End Testing**: Implement Cypress for full user workflow testing
2. **Performance Monitoring**: Add monitoring for page load times and API response times
3. **Error Boundary**: Implement React error boundaries for better error handling
4. **Accessibility**: Add ARIA labels and keyboard navigation improvements
5. **SEO Optimization**: Add proper meta tags and structured data

Testing Commands

```
# Run all tests
npm test

# Run tests in watch mode
npm run test:watch

# Run tests with coverage
npm run test:coverage

# Run integration tests only
npm run test:integration
```

Verification Steps

1. **Authentication Test**:
   - Visit any protected route → Should redirect to login
   - Login with admin/adminadmin → Should access protected routes

2. **System Design Test**:
   - Visit `/system-design-questions` → Should show 22 questions
   - Visit `/system-design-strategy` → Should show frameworks without errors

3. **Navigation Test**:
   - Resize browser window → Navigation should remain usable
   - Test mobile view → Should show hamburger menu

4. **API Test**:
   `bash`
   `curl http://localhost:3000/api/system-design-frameworks`
   `curl http://localhost:3000/api/system-design-questions`

Database Schema Compliance

All database operations now use correct snake_case field names:
- `key_principles` instead of `keyPrinciples`
- `use_cases` instead of `useCases`
- `architecture_focus` instead of `architectureFocus`
- `complexity_level` instead of `complexityLevel`
- And all other fields properly mapped

The application is now fully functional with all critical issues resolved and a comprehensive testing framework in place.