Static Code Analysis Checklist

- SECURITY CHECKS

Authentication & Authorization

- [ ] **JWT Secret**: Is `JWT_SECRET` in `.env` and NOT hardcoded?
- [ ] **Password Hashing**: Are all passwords hashed with Argon2 (never plain text)?
- [ ] **Token Expiration**: Are JWT tokens set to expire (check `createToken.js`)?
- [ ] **Authorization Middleware**: Is `checkAuth` middleware applied to all protected routes?
- [ ] **Input Validation**: Are all user inputs validated (username, password)?
- [ ] **SQL Injection**: Are database queries using parameterized queries (`$1`, `$2`)?

Environment Variables

- [ ] **`.env` in `.gitignore`**: Ensure `.env` files are NOT committed to Git
- [ ] **No Secrets in Code**: No passwords/secrets hardcoded in source files
- [ ] **DATABASE_URL**: Database credentials in `.env` only

CODE QUALITY CHECKS

Backend (`/backend`)

Routes

- [ ] **Error Handling**: All routes wrapped in `try...catch` blocks?
- [ ] **Status Codes**: Correct HTTP status codes used (200, 401, 500)?
- [ ] **Response Format**: Consistent JSON response structure?
- [ ] **Route Naming**: RESTful naming conventions followed?

Database Queries (`/backend/querie`)

- [ ] **Connection Pooling**: Using connection pool (not creating new connections)?
- [ ] **Query Errors**: Database errors are caught and handled?
- [ ] **Return Values**: Functions return consistent data types?
- [ ] **Null Checks**: Handle cases when user not found (`return null`)?

Middleware

- [ ] **Next() Called**: Middleware calls `next()` when successful?
- [ ] **Error Responses**: Middleware sends proper error responses?
- [ ] **Token Validation**: JWT verification errors are caught?

Frontend

Components

- [ ] **State Management**: Using `useState` appropriately?
- [ ] **Effect Cleanup**: `useEffect` has proper dependencies?
- [ ] **Error Handling**: API errors displayed to user?
- [ ] **Loading States**: Show loading indicators during async operations?

- FILE & STRUCTURE CHECKS

General

- [ ] **File Naming**: Consistent naming convention (camelCase/kebab-case)?
- [ ] **Imports**: No unused imports in files?
- [ ] **Console Logs**: Remove/comment debug `console.log()` statements?
- [ ] **TODO Comments**: All `TODO` comments addressed or documented?

Backend Structure

```
✓ Routes in /routes
✓ Database logic in /querie
✓ Middleware in /middleware
✓ Utilities in /utils
```

Frontend Structure

```
✓ Components in /src/comps
✓ Assets in /src/comps/assets
✓ Styles alongside components
```

- CONFIGURATION CHECKS

Backend Configuration

- [ ] **package.json**: All dependencies used? No unused packages?
- [ ] **Port Configuration**: Port configurable via environment variable?
- [ ] **Environment**: Separate configs for dev/prod?

Frontend Configuration

- [ ] **package.json**: Scripts working (`start`, `build`, `test`)?
- [ ] **API URL**: Backend URL configurable (not hardcoded)?
- [ ] **Build Configuration**: Production build working?

Database Configuration

- [ ] **init.sql**: Contains all necessary tables and data?
- [ ] **Constraints**: Primary keys, unique constraints, foreign keys defined?
- [ ] **Test Data**: Test users with known passwords for testing?

- DOCUMENTATION CHECKS

Code Documentation

- [ ] **Function Comments**: Complex functions have explanatory comments?
- [ ] **API Documentation**: Endpoint purposes clear?
- [ ] **Setup Instructions**: README has setup steps?

Code Readability

- [ ] **Variable Names**: Descriptive variable names?
- [ ] **Magic Numbers**: Constants defined for magic numbers?
- [ ] **Function Length**: Functions under 50 lines (split if longer)?