TechCorp Solutions

Web Programming 2 - Final Project Documentation

Mikhael Nabil Salama Rezk

Szabó Lilla

November 28, 2025

TechCorp Solutions - Web Application Documentation

Project Name: TechCorp Solutions Web Application
Course: Web Programming 2
Date: November 28, 2025

Developers:

- **Mikhael Nabil Salama Rezk**
- **Szabó Lilla** (Instructor/Colleague)

Table of Contents

---

1. Project Overview

TechCorp Solutions is a full-stack Node.js web application deployed on a Linux server. The application demonstrates modern web development practices including user authentication, CRUD operations, database integration, and administrative features.

Key Technologies

- **Backend:** Node.js v22.19.0, Express.js 4.x
- **Frontend:** EJS Templates, Bootstrap 5 (Lux Theme)
- **Database:** MySQL/MariaDB with Mock Mode fallback
- **Process Manager:** PM2 (Production deployment)
- **Version Control:** Git/GitHub
- **Server:** Ubuntu Linux (143.47.98.96)

Application URL

Live Application: http://143.47.98.96/app206/

---

2. Technical Architecture

Application Structure

```
exercise/
├── start.js                # Main application entry point
├── package.json            # Dependencies and scripts
├── ecosystem.config.js     # PM2 configuration
├── routes/                 # Route handlers
│   ├── auth.js             # Authentication routes
│   ├── database.js         # Database display
│   ├── contact.js          # Contact form
│   ├── messages.js         # Message management
│   ├── crud.js             # Product CRUD
│   └── projects.js         # Project management
├── views/                  # EJS templates
│   ├── layout.ejs          # Main layout template
│   ├── index.ejs           # Homepage
│   ├── login.ejs           # Login page
│   ├── register.ejs        # Registration page
│   ├── contact.ejs         # Contact form
│   ├── messages.ejs        # Messages list
│   ├── message_detail.ejs  # Message details
│   ├── message_edit.ejs    # Edit message
│   ├── message_reply.ejs   # Reply to message
│   ├── crud_products.ejs   # Products list
│   ├── crud_product_form.ejs # Add/Edit product
│   ├── projects.ejs        # Projects list
│   ├── project_form.ejs    # Add/Edit project
│   └── project_view.ejs    # Project details
└── public/                 # Static assets
    └── css/                # Stylesheets
```

Environment Configuration

- **BASE_PATH:** `/app206` (Nginx reverse proxy path)
- **INTERNAL_PORT:** `4206` (Application listening port)
- **NODE_ENV:** `production`
- **Database:** Mock mode (studb206 user not created)

---

3. Deployment Information

Server Details

- **Server IP:** 143.47.98.96
- **Username:** student206
- **Application Directory:** `~/webprogramming_with_lilla/`
- **PM2 Process Name:** app206
- **PM2 Binary Location:** `~/.local/bin/pm2`

Deployment Process

1. **Git Push:** Changes pushed to GitHub repository
2. **SSH Connection:** Connect to server as student206
3. **Git Pull:** Update code from repository
4. **PM2 Restart:** Restart application with zero downtime

PM2 Configuration (ecosystem.config.js)

```
module.exports = {
  apps: [{
    name: 'app206',
    script: './start.js',
    instances: 1,
    exec_mode: 'cluster',
    env: {
      NODE_ENV: 'production',
      BASE_PATH: '/app206',
      INTERNAL_PORT: 4206,
      DB_HOST: 'localhost',
      DB_USER: 'studb206',
      DB_PASSWORD: '[PASSWORD]',
      DB_NAME: 'studb206',
      SESSION_SECRET: '[SECRET]'
    }
  }]
};
```

---

4. Features Implementation

Homework Requirements (11 Features)

1. Homepage ✅

- Route: `/app206/`
- Modern landing page with hero section
- Service cards (Web Development, Mobile Apps, Cloud Solutions)
- Statistics counter
- Call-to-action sections

2. Products Page ✅

- Route: `/app206/products`
- Displays products from database
- Shows product details (name, description, price)
- Category filtering
- Joined with categories table

3. Database Page ✅

- Route: `/app206/database`

- Displays first 10 products with categories
- SQL JOIN demonstration
- Table format with Bootstrap styling

4. About Page ✅

- Route: `/app206/about`
- Company information
- Team member profiles
- Mission and values

5. Contact Page ✅

- Route: `/app206/contact`
- Contact form with validation
- Saves messages to database
- Success/error feedback

6. User Registration ✅

- Route: `/app206/register`
- Form with validation
- Password hashing (SHA-256)
- Duplicate username prevention

7. User Login ✅

- Route: `/app206/login`
- Passport.js authentication
- Session management
- Role-based access (admin/user)

8. User Logout ✅

- Route: `/app206/logout`
- Session destruction
- Redirect to homepage

9. Protected Route ✅

- Route: `/app206/protected`
- Requires authentication
- Displays user information
- Admin badge for administrators

10. Admin Route ✅

- Route: `/app206/admin`
- Admin-only access
- User management interface

- System statistics

11. Product CRUD ✅

- Route: `/app206/crud`
- Full Create, Read, Update, Delete
- Admin-only access
- Product management interface

Additional Features Implemented

12. Message Management ✅

- Route: `/app206/messages`
- View all contact messages
- Status management (new/read/replied/archived)
- Edit, delete, reply functionality
- Search and filter capabilities

13. Project Management ✅

- Route: `/app206/projects`
- Full CRUD for portfolio projects
- Image URL support
- Technology tags
- Project URL linking

---

5. Database Structure

Mock Database Implementation

Due to database user not being created, the application runs in **DEMO MODE** with in-memory mock data.

Tables & Structure

users

```
CREATE TABLE users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100),
    password_hash VARCHAR(255) NOT NULL,
    role VARCHAR(20) DEFAULT 'user',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

products

```
CREATE TABLE products (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    price DECIMAL(10,2),
    category_id INT,
    status VARCHAR(20) DEFAULT 'active',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

categories

```
CREATE TABLE categories (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    description TEXT
);
```

contact_messages

```
CREATE TABLE contact_messages (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    subject VARCHAR(200),
    message TEXT NOT NULL,
    status VARCHAR(20) DEFAULT 'new',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

projects

```
CREATE TABLE projects (
    id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(200) NOT NULL,
    description TEXT,
    technologies TEXT,
    image_url VARCHAR(500),
    project_url VARCHAR(500),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Mock Data

Default Users:

- **Admin:** username: `admin`, password: `admin123`, role: `admin`
- **Test User:** username: `testuser`, password: `hello`, role: `user`

Sample Products:

- Premium Hosting Plan ($29.99)
- Custom Web Development ($999.99)
- Mobile App Package ($1499.99)

Sample Projects:

- E-Commerce Platform (React, Node.js, MongoDB)
- Mobile Banking App (React Native, Firebase)

---

6. API Routes

Authentication Routes (`/auth`)

Method	Route	Description	Access
GET	`/login`	Login page	Public
POST	`/login`	Login handler	Public
GET	`/register`	Registration page	Public
POST	`/register`	Registration handler	Public
GET	`/logout`	Logout user	Authenticated

Product CRUD Routes (`/crud`)

Method	Route	Description	Access
GET	`/`	List products	Admin
GET	`/add`	Add product form	Admin
POST	`/add`	Create product	Admin
GET	`/view/:id`	View product	Admin
GET	`/edit/:id`	Edit product form	Admin
POST	`/edit/:id`	Update product	Admin
POST	`/delete/:id`	Delete product	Admin

Message Routes (`/messages`)

Method	Route	Description	Access
GET	`/`	List messages	Admin
GET	`/:id`	View message	Admin
POST	`/:id/status`	Update status	Admin
GET	`/:id/edit`	Edit message form	Admin
POST	`/:id/edit`	Update message	Admin
POST	`/:id/delete`	Delete message	Admin
GET	`/:id/reply`	Reply form	Admin
POST	`/:id/reply`	Send reply	Admin

Project Routes (`/projects`)

Method	Route	Description	Access
GET	`/`	List projects	Admin
GET	`/add`	Add project form	Admin
POST	`/add`	Create project	Admin
GET	`/view/:id`	View project	Admin
GET	`/edit/:id`	Edit project form	Admin
POST	`/edit/:id`	Update project	Admin
POST	`/delete/:id`	Delete project	Admin

Public Routes

Method	Route	Description	Access
GET	`/`	Homepage	Public
GET	`/products`	Products page	Public
GET	`/about`	About page	Public
GET	`/database`	Database display	Public
GET	`/contact`	Contact form	Public
POST	`/contact`	Submit message	Public
GET	`/protected`	Protected page	Authenticated
GET	`/admin`	Admin page	Admin

---

7. User Authentication

Implementation Details

Authentication Library: Passport.js with Local Strategy

Session Management:

- Express-session with MemoryStore
- Session secret from environment variables
- Cookie settings: secure in production

Password Security:

- SHA-256 hashing algorithm
- Salt added during registration
- Never stored in plain text

Authentication Flow

```
1. User submits login form
   ↓
2. Passport LocalStrategy validates credentials
   ↓
3. Mock database checks username/password
   ↓
4. Session created with user data
   ↓
5. User redirected to protected area
```

Middleware

isAuthenticated:

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

isAdmin:

```
function isAdmin(req, res, next) {
    if (req.isAuthenticated() && req.user.role === 'admin') {
        return next();
    }
    res.status(403).send('Admin access required');
}
```

---

8. Admin Panel

Admin Features

Product Management (`/crud`)

- **Create:** Add new products with details
- **Read:** View product list with categories
- **Update:** Edit product information
- **Delete:** Remove products with confirmation
- **Search:** Filter products by name/category
- **Status:** Toggle active/inactive status
- **Statistics:** Total, active, inactive counts

Message Management (`/messages`)

- **View All:** List contact messages
- **Status Updates:** Mark as new/read/replied/archived
- **Reply:** Respond to customer inquiries
- **Edit:** Modify message content
- **Delete:** Remove spam/invalid messages
- **Search:** Filter by name, email, subject
- **Details:** View full message content

Project Management (`/projects`)

- **Add Projects:** Create portfolio items
- **Edit Details:** Update project information
- **Delete Projects:** Remove old projects
- **View Details:** Full project display
- **Image Support:** Project screenshots
- **Technology Tags:** Tech stack display
- **External Links:** Link to live projects

Admin Navigation

- Dropdown menu in header (Admin icon)
- Quick access to:
    - Manage Products
    - Manage Projects
    - View Messages

---

9. Testing & Validation

Tested Functionality

✅ Authentication

- [x] User registration with duplicate prevention
- [x] User login with correct credentials
- [x] Login failure with incorrect credentials
- [x] Session persistence across pages
- [x] Logout functionality
- [x] Protected route access control
- [x] Admin-only route restrictions

✅ CRUD Operations

- [x] Create new products
- [x] Read/view product list
- [x] Update product details
- [x] Delete products with confirmation
- [x] All operations respect BASE_PATH
- [x] Proper error handling

✅ Message System

- [x] Contact form submission
- [x] Message storage in mock database
- [x] Admin message viewing
- [x] Status updates (AJAX)
- [x] Message editing
- [x] Message deletion
- [x] Reply functionality

✅ Project Management

- [x] Create new projects
- [x] View project list
- [x] Edit project details
- [x] Delete projects
- [x] Project detail view
- [x] Image and link support

✅ BASE_PATH Routing

- [x] All routes use `/app206` prefix
- [x] Redirects include BASE_PATH
- [x] Static assets load correctly
- [x] Forms submit to correct paths
- [x] AJAX calls include BASE_PATH

Browser Compatibility

- [x] Chrome/Edge (Latest)

- ✅ Firefox (Latest)
- ✅ Safari (Latest)
- ✅ Mobile browsers

Security Features

- ✅ Password hashing
- ✅ Session management
- ✅ Role-based access control
- ✅ Input validation
- ✅ XSS prevention (EJS escaping)
- ✅ CSRF consideration

---

10. Conclusion

Project Achievements

This project successfully demonstrates:

1. **Full-Stack Development:** Complete Node.js/Express application with frontend and backend integration
2. **Database Operations:** CRUD operations with MySQL (mock mode implementation)
3. **User Authentication:** Secure login/registration with role-based access
4. **Modern UI/UX:** Bootstrap 5 with Lux theme for professional appearance
5. **Production Deployment:** Live application on Linux server with PM2
6. **Git Workflow:** Version control with GitHub integration
7. **Admin Features:** Comprehensive management interfaces
8. **API Design:** RESTful routes with proper HTTP methods

Technical Skills Demonstrated

- **Backend:** Node.js, Express.js, routing, middleware
- **Frontend:** EJS templating, Bootstrap 5, responsive design
- **Database:** MySQL queries, JOINs, mock data implementation
- **Security:** Authentication, authorization, password hashing
- **DevOps:** Linux server management, PM2, Git deployment
- **JavaScript:** ES6+, async/await, fetch API, DOM manipulation

Future Enhancements

1. **Database Connection:** Enable real MySQL database (requires admin to create studb206 user)
2. **Email Integration:** Implement actual email sending for replies
3. **File Uploads:** Add image upload for products and projects
4. **API Endpoints:** Create RESTful API for external integrations
5. **Search Optimization:** Full-text search across all content

6. **Pagination:** Implement pagination for large datasets
7. **Export Features:** PDF/CSV export for messages and products
8. **Activity Logs:** Track admin actions and user activities

Lessons Learned

1. **BASE_PATH Handling:** Critical importance of consistent path management in reverse proxy scenarios
2. **Mock Database:** Effective fallback strategy when database access is unavailable
3. **EJS Templating:** Proper syntax for avoiding errors with embedded JavaScript
4. **Data Attributes:** Safer approach than inline onclick handlers with dynamic content
5. **PM2 Deployment:** Zero-downtime deployments with process management
6. **Git Workflow:** Importance of version control for production deployments

---

Appendix

Deployment Commands

Deploy to Server:

```
git add -A
git commit -m "Description of changes"
git push origin main
ssh student206@143.47.98.96 "cd ~/webprogramming_with_lilla && git pull && ~/.local/bin/pm2 restart app206
```

Check Application Status:

```
ssh student206@143.47.98.96 "~/.local/bin/pm2 status"
```

View Logs:

```
ssh student206@143.47.98.96 "~/.local/bin/pm2 logs app206 --lines 50"
```

Test Credentials

Admin Account:

- Username: `admin`
- Password: `admin123`

Regular User:

- Username: `testuser`
- Password: `hello`

Important URLs

- **Homepage:** http://143.47.98.96/app206/
- **Login:** http://143.47.98.96/app206/login

- **Products:** http://143.47.98.96/app206/products
- **Contact:** http://143.47.98.96/app206/contact
- **Admin Panel:** http://143.47.98.96/app206/admin
- **Product CRUD:** http://143.47.98.96/app206/crud
- **Messages:** http://143.47.98.96/app206/messages
- **Projects:** http://143.47.98.96/app206/projects

Document Version: 1.0
Last Updated: November 28, 2025
Status: Production Ready ✅
