Hostel Management System - Software Requirements Specification (SRS)

Project Overview

A web-based hostel management system for managing student accommodations, room allocations, complaints, payments, and visitor tracking.

Target Users: Hostel administrators, wardens, and students

Tech Stack:

- **Frontend:** React, Tailwind CSS, shadcn/ui, React Router, React Hook Form, Axios
- **Backend:** Node.js, Express.js, NeonDB (PostgreSQL)
- **Authentication:** JWT (JSON Web Tokens)
- **Additional Tools:** Bcrypt (password hashing), Zod (validation), date-fns (date handling)

Development Phases (Beginner-Friendly Hierarchy)

🟢 PHASE 1: Foundation & Setup (Week 1-2)

Difficulty: Easy | **Focus:** Getting comfortable with the environment

Learning Goals:

- Understand project structure
- Set up development environment
- Create basic UI components

Tasks:

1. **Environment Setup**
   - Install Node.js, VS Code, Git
   - Create GitHub repository
   - Initialize React project with Vite
   - Set up Tailwind CSS and shadcn/ui
   - Create basic folder structure

2. **Database Setup**
    - Create NeonDB account
    - Design basic database schema (on paper first)
    - Set up connection
3. **Basic Pages (Static)**
    - Landing page with hero section
    - Login page UI
    - Registration page UI
    - Dashboard layout (empty for now)

Recommended Libraries for Phase 1:

- `react-router-dom` - Page navigation
- `lucide-react` - Icons (comes with shadcn)

🟡 PHASE 2: Authentication System (Week 3-4)

Difficulty: Medium | **Focus:** User management basics

Learning Goals:

- Understand REST APIs
- Learn JWT authentication
- Connect frontend to backend

Features to Build:

Module 1: User Authentication

- User registration (students only for now)
- User login with email/password
- JWT token generation and storage
- Protected routes (redirect if not logged in)
- Logout functionality

Module 2: User Roles

- Three roles: Admin, Warden, Student
- Role-based access control
- Different dashboards for each role

Database Tables:

```
users
- id (primary key)
- name
- email (unique)
- password (hashed)
- phone
- role (admin/warden/student)
- created_at
- updated_at
```

Recommended Libraries:

- **Backend:** `jsonwebtoken`, `bcryptjs`, `express-validator`
- **Frontend:** `axios`, `react-hook-form`, `zod`

🟡 PHASE 3: Room Management (Week 5-6)

Difficulty: Medium | **Focus:** CRUD operations

Learning Goals:

- Create, Read, Update, Delete (CRUD)
- Working with forms
- Display data in tables

Features to Build:

Module 3: Room Management (Admin/Warden)

- Add new rooms (room number, floor, capacity, type)
- View all rooms in a table
- Edit room details

- Delete rooms
- Filter rooms by floor/availability
- View room occupancy status

Module 4: Room Allocation

- Assign students to rooms
- Check available beds
- View current occupants
- Remove student from room
- Room transfer functionality

Database Tables:

```
rooms
- id
- room_number (unique)
- floor
- capacity (total beds)
- occupied (current occupants)
- room_type (single/double/triple/dormitory)
- status (available/full/maintenance)

room_allocations
- id
- student_id (foreign key)
- room_id (foreign key)
- allocated_date
- checkout_date
- status (active/inactive)
```

Recommended UI Components:

- Data tables with shadcn Table component
- Dialog/Modal for forms
- Select dropdowns
- Search and filter functionality

🟠 PHASE 4: Student Profile & Dashboard (Week 7-8)

Difficulty: Medium-Hard | **Focus:** User-specific data

Features to Build:

Module 5: Student Profile

- View personal information
- Edit profile details
- Upload profile picture (optional - can use placeholder initially)
- View allocated room details
- Emergency contact information

Module 6: Student Dashboard

- Quick stats (room number, floor, roommates)
- Recent complaints
- Payment history
- Upcoming dues
- Announcements board

Database Tables:

```
student_profiles
- id
- user_id (foreign key)
- guardian_name
- guardian_phone
- address
- emergency_contact
- date_of_birth
- blood_group
```

Recommended Libraries:

- `date-fns` - Date formatting and manipulation
- `recharts` - Simple charts for dashboard (optional)

🟠 PHASE 5: Complaint Management System (Week 9-10)

Difficulty: Medium-Hard | **Focus:** Status workflows

Features to Build:

Module 7: Complaints (Student Side)

- Submit new complaint with category
- Add description and room number
- View complaint status (pending/in-progress/resolved)
- View complaint history
- Add comments to existing complaints

Module 8: Complaint Resolution (Admin/Warden Side)

- View all complaints
- Filter by status/category/priority
- Update complaint status
- Assign priority levels
- Add resolution notes
- Mark as resolved

Database Tables:

```
complaints
- id
- student_id (foreign key)
- room_id (foreign key)
- category (maintenance/electrical/plumbing/housekeeping/other)
- title
- description
- priority (low/medium/high)
- status (pending/in-progress/resolved)
- created_at
- resolved_at

complaint_comments
- id
- complaint_id (foreign key)
- user_id (foreign key)
- comment
- created_at
```

UI Features:

- Status badges with colors
- Timeline view for complaint history
- Filter and sort options
- Priority indicators

🔴 PHASE 6: Payment & Fee Management (Week 11-12)

Difficulty: Hard | **Focus:** Financial records

Features to Build:

Module 9: Fee Structure (Admin)

- Define monthly/semester fees
- Set room type specific charges
- Add additional charges (electricity, water, etc.)
- Late payment penalties

- Fee waivers/discounts

Module 10: Payment Tracking (All Users)

- Record manual payments (initially - no payment gateway)
- View payment history
- Generate receipts (PDF - optional for later)
- Payment due notifications
- Outstanding balance display

Module 11: Reports (Admin/Warden)

- Monthly collection reports
- Defaulter list
- Room-wise revenue
- Export to CSV

Database Tables:

```
fee_structure
- id
- room_type
- base_fee
- electricity_charge
- water_charge
- maintenance_charge
- total_monthly_fee

payments
- id
- student_id (foreign key)
- amount
- payment_date
- payment_mode (cash/online/card)
- month_year
- status (pending/paid)
- receipt_number
- remarks

payment_dues
- id
- student_id (foreign key)
- month_year
- due_amount
- paid_amount
- due_date
- status (pending/overdue/paid)
```

Recommended Libraries:

- `react-to-print` - Print receipts
- `papaparse` - CSV export

🔴 PHASE 7: Visitor Management (Week 13-14)

Difficulty: Hard | **Focus:** Real-time tracking

Features to Build:

Module 12: Visitor Entry (Security/Warden)

- Record visitor entry
- Capture visitor details (name, phone, purpose)
- Photo ID type and number
- Whom to meet (student)
- Entry time auto-recorded
- Record exit time

Module 13: Visitor Logs

- View current visitors in hostel
- Visitor history with filters
- Search by visitor name or student
- Daily/monthly visitor reports
- Generate visitor passes (optional)

Database Tables:

```
visitors
- id
- visitor_name
- visitor_phone
- id_proof_type
- id_proof_number
- student_id (foreign key - whom to meet)
- purpose
- entry_time
- exit_time
- recorded_by (user_id)
```

UI Features:

- Quick entry form for security
- Real-time visitor count
- Active visitor list
- Search and filter options

🟣 PHASE 8: Additional Features (Week 15-16)

Difficulty: Variable | **Focus:** Enhancement

Module 14: Announcements & Notices

- Admin can post announcements
- Display on student dashboard
- Mark as important/urgent
- Archive old announcements

Module 15: Attendance/In-Out Register (Optional)

- Daily check-in/check-out for students
- Late arrival tracking
- Generate attendance reports

Module 16: Mess Management (Optional - Future Scope)

- Meal plan subscriptions
- Mess fee management
- Menu display

Database Tables:

```
announcements
- id
- title
- description
- priority (normal/important/urgent)
- posted_by (user_id)
- created_at
- expires_at
```

Additional Recommendations

Essential NPM Packages

Frontend:

```
- react-router-dom: Navigation
- axios: HTTP requests
- react-hook-form: Form handling
- zod: Schema validation
- date-fns: Date utilities
- react-hot-toast: Notifications
- @tanstack/react-query: Server state management (recommended for advanced phase)
```

Backend:

```
- express: Web framework
- @neondatabase/serverless: NeonDB client
- jsonwebtoken: JWT authentication
- bcryptjs: Password hashing
- express-validator: Input validation
- dotenv: Environment variables
- cors: Cross-origin requests
- helmet: Security headers
- morgan: HTTP request logger
```

Development Best Practices

1. **Git Workflow:**
    - Create separate branches for each feature
    - Commit frequently with clear messages
    - Use `.gitignore` for node_modules and .env
2. **Code Organization:**

```
frontend/
├── src/
│   ├── components/    # Reusable components
│   ├── pages/         # Page components
│   ├── services/      # API calls
│   ├── utils/         # Helper functions
│   ├── hooks/         # Custom hooks
│   └── context/       # State management

backend/
├── controllers/       # Request handlers
├── models/            # Database models
├── routes/            # API routes
├── middleware/        # Auth, validation
└── config/            # Database config
```

3. **Security Considerations:**
   - Never commit .env files
   - Hash all passwords with bcrypt
   - Use JWT for authentication
   - Validate all inputs on backend
   - Use HTTPS in production
   - Implement rate limiting for API
4. **Testing (Later Phases):**
   - Use Postman for API testing
   - Consider Jest for unit tests
   - Manual testing checklist for each feature

Learning Resources Recommended

- **React:** Official React docs (react.dev)
- **Tailwind CSS:** Official Tailwind docs
- **shadcn/ui:** shadcn component documentation
- **Node.js:** Node.js official guides
- **PostgreSQL:** PostgreSQL tutorial

- **Authentication:** JWT.io for understanding tokens

Deployment Strategy (After Completion)

Frontend: Vercel or Netlify (Free tier) **Backend:** Render or Railway (Free tier) **Database:** NeonDB (Already cloud-based)

Success Metrics

Phase 1-2 Complete:

✅ User can register and login
✅ Different dashboards for different roles

Phase 3-4 Complete:

✅ Admin can manage rooms
✅ Students can view their allocated room
✅ Student profile is functional

Phase 5-6 Complete:

✅ Students can raise complaints
✅ Admin can track and resolve complaints
✅ Payment records are maintained

Phase 7-8 Complete:

✅ Visitor tracking is functional
✅ All reports are generating correctly
✅ System is fully functional

Timeline Summary

- **Total Duration:** 16 weeks (4 months)
- **Daily Commitment:** 2-3 hours
- **Weekly Goals:** Complete 1 module/week
- **Buffer Time:** Built into each phase

Next Steps

1. ✅ Read this document thoroughly
2. ✅ Set up your development environment
3. ✅ Create a GitHub repository
4. ✅ Start with Phase 1, Module by Module
5. ✅ Don't rush - understand each concept
6. ✅ Build one feature completely before moving to next
7. ✅ Test each feature after building

Notes for Beginners

- **Don't skip phases:** Each builds on the previous one
- **Google is your friend:** You'll encounter errors - that's normal
- **Take breaks:** Coding fatigue is real
- **Ask for help:** Use developer communities (Stack Overflow, Reddit)
- **Version control:** Commit your code regularly
- **Start simple:** Don't add complex features initially
- **UI can be improved later:** Focus on functionality first
- **One thing at a time:** Complete backend API, then connect frontend

Remember: Every expert was once a beginner. Take it step by step, and you'll have an impressive portfolio project! 🚀

Good luck with your project!