Dayflow HRMS – Module-Wise Development with Pages & APIs (MERN)

🧱 MODULE 0 – Project Setup (Foundation)

Pages

- No UI pages (setup only)

APIs

- No APIs

Work

- Backend + Frontend setup
- MongoDB connection
- Environment configuration

🔐 MODULE 1 – Authentication & Authorization

Pages

1. **Sign In Page**
2. **Sign Up Page** (Admin/HR only)

APIs

Method	API
POST	/api/auth/login
POST	/api/auth/logout
GET	/api/auth/me
POST	/api/auth/change-password
POST	/api/auth/first-login-reset

Access

- Employee & Admin (Login)
- Only Admin can create users

👥 MODULE 2 – Employee Management

Pages

3. **Employees List Page (Admin view)**

4. **Employee Self Page (Employee view)**
5. **Employee Detail / Profile Page**

APIs

Method	API
POST	/api/employees
GET	/api/employees
GET	/api/employees/:id
PUT	/api/employees/:id
DELETE	/api/employees/:id
GET	/api/employees/self
POST	/api/employees/upload-photo

Access

- Admin: full access
- Employee: self-view only

👤 **MODULE 3 – Profile Management**

Pages

6. **Resume / About Page**
7. **Private Information Page**
8. **Security Page**

(Tabs inside My Profile)

APIs

Method	API
PUT	/api/profile/private-info
PUT	/api/profile/skills
PUT	/api/profile/certifications
PUT	/api/profile/security

Access

- Employee: own profile

- Admin: any employee profile

⏱️ **MODULE 4 – Attendance Management**

Pages

9. **Attendance Page – Employee View**
10. **Attendance Page – Admin/HR View**

APIs

Method	API
POST	/api/attendance/check-in
POST	/api/attendance/check-out
GET	/api/attendance/self
GET	/api/attendance/all
GET	/api/attendance/date
PUT	/api/attendance/correct

Access

- Employee: own attendance
- Admin/HR: all attendance

🏖️ **MODULE 5 – Time Off / Leave Management**

Pages

11. **Time Off Page – Employee View**
12. **Time Off Page – Admin/HR View**

(Same page, role-based behavior)

APIs

Method	API
POST	/api/leaves/apply
GET	/api/leaves/self
GET	/api/leaves/all
PUT	/api/leaves/approve/:id

Method	API
PUT	/api/leaves/reject/:id
POST	/api/leaves/upload-proof

Access

- Employee: apply & view
- Admin/HR: approve/reject

💰 MODULE 6 – Salary & Payroll Management

Pages

13. **Salary Information Page** *(Admin only tab)*

APIs

Method	API
POST	/api/salary/create
GET	/api/salary/:empId
PUT	/api/salary/update/:empId
GET	/api/salary/all
GET	/api/salary/calculate/:empId

Access

- Admin/HR only

📊 MODULE 7 – Dashboard & Status Indicators

Pages

14. **Dashboard Page**

APIs

Method	API
GET	/api/dashboard/stats

Access

- Employee: limited stats
- Admin: full stats

🔒 MODULE 8 – Security & Role Control

Pages

15. **Unauthorized / Access Denied Page**

APIs

Method	API
GET	/api/system/config

Access

- System-level

⚙️ MODULE 9 – Utility & Uploads

Pages

- Used internally (no separate page)

APIs

Method	API
POST	/api/upload/logo
POST	/api/upload/document

✅ FINAL COUNT SUMMARY

📄 **Pages**

- **15 Pages**

🔌 **APIs**

- **36 REST APIs**

🎯 How to Develop (Golden Rule)

Finish **one module completely** (pages + APIs + testing)
before starting the next module.

1. User Schema (Authentication & Roles)

📁 models/User.js

```
import mongoose from "mongoose";

const userSchema = new mongoose.Schema({
 loginId: {
  type: String,
  unique: true,
  required: true
 },

 email: {
  type: String,
  unique: true,
  required: true
 },

 password: {
  type: String,
  required: true
 },

 role: {
  type: String,
  enum: ["EMPLOYEE", "ADMIN"],
  default: "EMPLOYEE"
 },

 isFirstLogin: {
  type: Boolean,
  default: true
```

```
  },

  isActive: {
    type: Boolean,
    default: true
  }

}, { timestamps: true });

export default mongoose.model("User", userSchema);
```

👥 2. Employee Schema (Employee Management)

📁 models/Employee.js

```
import mongoose from "mongoose";

const employeeSchema = new mongoose.Schema({
  userId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "User",
    required: true
  },

  employeeCode: {
    type: String,
    unique: true,
    required: true
  },

  fullName: String,
  phone: String,
  address: String,
```

```
  designation: String,
  department: String,
  dateOfJoining: Date,

  profileImage: String,

  status: {
    type: String,
    enum: ["ACTIVE", "INACTIVE"],
    default: "ACTIVE"
  }

}, { timestamps: true });

export default mongoose.model("Employee", employeeSchema);
```

👤 3. Profile Schema (Resume / Private Info)

📁 models/Profile.js

```
import mongoose from "mongoose";

const profileSchema = new mongoose.Schema({
  employeeId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Employee",
    required: true
  },

  skills: [String],
  certifications: [String],
  experience: String,
```

```
  emergencyContact: {
    name: String,
    phone: String
  }

}, { timestamps: true });

export default mongoose.model("Profile", profileSchema);
```

⏱️ **4. Attendance Schema**

📁 models/Attendance.js

```
import mongoose from "mongoose";

const attendanceSchema = new mongoose.Schema({
  employeeId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Employee",
    required: true
  },

  date: {
    type: Date,
    required: true
  },

  checkIn: Date,
  checkOut: Date,

  workHours: Number,
  extraHours: Number,
```

```
  status: {
    type: String,
    enum: ["PRESENT", "ABSENT", "HALF_DAY", "LEAVE"],
    default: "PRESENT"
  }

}, { timestamps: true });

export default mongoose.model("Attendance", attendanceSchema);
```

🏖️ 5. Leave / Time-Off Schema

📁 models/Leave.js

```
import mongoose from "mongoose";

const leaveSchema = new mongoose.Schema({
  employeeId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Employee",
    required: true
  },

  leaveType: {
    type: String,
    enum: ["PAID", "SICK", "UNPAID"],
    required: true
  },

  fromDate: Date,
  toDate: Date,
```

```
  totalDays: Number,
  reason: String,

  status: {
    type: String,
    enum: ["PENDING", "APPROVED", "REJECTED"],
    default: "PENDING"
  },

  adminComment: String,
  attachment: String

}, { timestamps: true });

export default mongoose.model("Leave", leaveSchema);
```

💰 **6. Salary Schema (Payroll)**

📁 models/Salary.js

```
import mongoose from "mongoose";

const salarySchema = new mongoose.Schema({
  employeeId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Employee",
    required: true
  },

  wageType: {
    type: String,
    enum: ["MONTHLY", "YEARLY"],
    required: true
```

```
  },

  baseSalary: Number,

  components: {
    basic: Number,
    hra: Number,
    allowance: Number,
    bonus: Number
  },

  deductions: {
    pf: Number,
    tax: Number,
    other: Number
  },

  totalSalary: Number

}, { timestamps: true });

export default mongoose.model("Salary", salarySchema);
```

🏢 7. Company Schema

📁 models/Company.js

```
import mongoose from "mongoose";

const companySchema = new mongoose.Schema({
  companyName: {
    type: String,
    required: true
```

```
  },

  companyCode: {
    type: String,
    unique: true,
    required: true
  },

  logo: String,

  createdBy: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "User"
  }

}, { timestamps: true });

export default mongoose.model("Company", companySchema);
```

📊 Schema Relationship Diagram (Textual)

```
User ——1——1—— Employee ——1——1—— Profile
  |                |
  |                ├— Attendance (many)
  |                ├— Leave (many)
  |                └— Salary (1)
  |
  └— Company
```