ePathshala System Flow Chart

Complete System Architecture Flow

```
graph TB
    %% User Interface Layer
    subgraph "Frontend (React + Material-UI)"
        UI[User Interface]
        Login[Login Page]
        Dashboard[Dashboard Pages]
        Chat[Chat Interface]
        Exam[Exam Interface]
        Assign[Assignment Interface]
        Calendar[Calendar Interface]
    end

    %% API Gateway Layer
    subgraph "API Gateway"
        Router[React Router]
        AuthGuard[Protected Routes]
        API[API Service Layer]
    end

    %% Backend Services Layer
    subgraph "Backend Services (Spring Boot)"
        AuthController[Auth Controller]
        TeacherController[Teacher Controller]
        StudentController[Student Controller]
        ParentController[Parent Controller]
        AdminController[Admin Controller]
        ChatController[Chat Controller]
        ExamController[Exam Controller]
        FileController[File Controller]
    end

    %% Business Logic Layer
    subgraph "Business Logic Layer"
        AuthService[Auth Service]
        TeacherService[Teacher Service]
        StudentService[Student Service]
        ParentService[Parent Service]
        AdminService[Admin Service]
        ChatService[Chat Service]
        ExamService[Exam Service]
        NotificationService[Notification Service]
    end

    %% Data Access Layer
    subgraph "Data Access Layer"
```

```
        UserRepo[User Repository]
        StudentRepo[Student Repository]
        TeacherRepo[Teacher Repository]
        AttendanceRepo[Attendance Repository]
        GradeRepo[Grade Repository]
        AssignmentRepo[Assignment Repository]
        ExamRepo[Exam Repository]
        ChatRepo[Chat Repository]
    end

    %% Database Layer
    subgraph "Database (MySQL)"
        Users[(Users Table)]
        Students[(Students Table)]
        Teachers[(Teachers Table)]
        Parents[(Parents Table)]
        Attendance[(Attendance Table)]
        Grades[(Grades Table)]
        Assignments[(Assignments Table)]
        Submissions[(Submissions Table)]
        Exams[(Exams Table)]
        Questions[(Questions Table)]
        ChatMessages[(Chat Messages Table)]
        Notifications[(Notifications Table)]
    end

    %% External Services
    subgraph "External Services"
        JWT[JWT Authentication]
        WebSocket[WebSocket Server]
        FileStorage[File Storage]
        EmailService[Email Service]
    end

    %% User Flow Connections
    UI --> Router
    Router --> AuthGuard
    AuthGuard --> API
    API --> AuthController
    API --> TeacherController
    API --> StudentController
    API --> ParentController
    API --> AdminController
    API --> ChatController
    API --> ExamController

    %% Controller to Service Connections
    AuthController --> AuthService
    TeacherController --> TeacherService
    StudentController --> StudentService
    ParentController --> ParentService
    AdminController --> AdminService
    ChatController --> ChatService
```

```
    ExamController --> ExamService

    %% Service to Repository Connections
    AuthService --> UserRepo
    TeacherService --> TeacherRepo
    TeacherService --> AttendanceRepo
    TeacherService --> GradeRepo
    TeacherService --> AssignmentRepo
    StudentService --> StudentRepo
    StudentService --> AssignmentRepo
    StudentService --> GradeRepo
    ParentService --> ParentRepo
    AdminService --> UserRepo
    ChatService --> ChatRepo
    ExamService --> ExamRepo

    %% Repository to Database Connections
    UserRepo --> Users
    StudentRepo --> Students
    TeacherRepo --> Teachers
    ParentRepo --> Parents
    AttendanceRepo --> Attendance
    GradeRepo --> Grades
    AssignmentRepo --> Assignments
    AssignmentRepo --> Submissions
    ExamRepo --> Exams
    ExamRepo --> Questions
    ChatRepo --> ChatMessages

    %% External Service Connections
    AuthService --> JWT
    ChatService --> WebSocket
    FileController --> FileStorage
    NotificationService --> EmailService

    %% Styling
    classDef frontend fill:#e1f5fe
    classDef backend fill:#f3e5f5
    classDef database fill:#e8f5e8
    classDef external fill:#fff3e0

    class UI,Login,Dashboard,Chat,Exam,Assign,Calendar,Router,AuthGuard,API
frontend
    class
AuthController,TeacherController,StudentController,ParentController,AdminContro
ller,ChatController,ExamController,FileController,AuthService,TeacherService,St
udentService,ParentService,AdminService,ChatService,ExamService,NotificationSer
vice,UserRepo,StudentRepo,TeacherRepo,AttendanceRepo,GradeRepo,AssignmentRepo,E
xamRepo,ChatRepo backend
    class
Users,Students,Teachers,Parents,Attendance,Grades,Assignments,Submissions,Exams
,Questions,ChatMessages,Notifications database
    class JWT,WebSocket,FileStorage,EmailService external
```

User Authentication Flow

```
sequenceDiagram
    participant U as User
    participant F as Frontend
    participant A as AuthController
    participant S as AuthService
    participant R as UserRepository
    participant D as Database
    participant J as JWT

    U->>F: Enter credentials
    F->>A: POST /api/auth/login
    A->>S: authenticateUser(credentials)
    S->>R: findByEmail(email)
    R->>D: SELECT * FROM users WHERE email = ?
    D-->>R: User data
    R-->>S: User object
    S->>S: validatePassword(password)
    S->>J: generateToken(user)
    J-->>S: JWT token
    S-->>A: LoginResponse(token, user)
    A-->>F: 200 OK + token
    F->>F: Store token in localStorage
    F-->>U: Redirect to dashboard
```

Teacher Dashboard Flow

```
flowchart TD
    A[Teacher Login] --> B[Teacher Dashboard]
    B --> C{Select Action}

    C -->|Mark Attendance| D[Attendance Page]
    C -->|Manage Grades| E[Grades Page]
    C -->|Create Assignment| F[Assignment Page]
    C -->|Handle Leave Requests| G[Leave Requests Page]
    C -->|Manage Exams| H[Exam Page]
    C -->|Online Classes| I[Online Classes Page]

    D --> J[Mark Student Attendance]
    E --> K[Enter Student Grades]
    F --> L[Create New Assignment]
    G --> M[Approve/Reject Leave]
    H --> N[Create/Manage Exams]
    I --> O[Schedule Online Class]

    J --> P[Save to Database]
```

```
    K --> P
    L --> P
    M --> P
    N --> P
    O --> P

    P --> Q[Update Dashboard]
    Q --> B
```

Student Dashboard Flow

```
flowchart TD
    A[Student Login] --> B[Student Dashboard]
    B --> C{Select Action}

    C -->|View Assignments| D[Assignments Page]
    C -->|Take Exams| E[Exams Page]
    C -->|Check Grades| F[Grades Page]
    C -->|View Attendance| G[Attendance Page]
    C -->|Submit Leave Request| H[Leave Request Page]
    C -->|View Calendar| I[Calendar Page]

    D --> J[View Assignment List]
    E --> K[Start Exam]
    F --> L[View Grade Report]
    G --> M[View Attendance Record]
    H --> N[Submit Leave Form]
    I --> O[View Academic Calendar]

    J --> P[Submit Assignment]
    K --> Q[Answer Questions]
    L --> R[Download Report]
    M --> S[Track Progress]
    N --> T[Wait for Approval]
    O --> U[View Events]

    P --> V[Upload File]
    Q --> W[Submit Exam]
    V --> X[Save to Database]
    W --> X
    X --> Y[Update Dashboard]
    Y --> B
```

Real-time Chat Flow

```
sequenceDiagram
    participant U1 as User 1
    participant F1 as Frontend 1
```

```
    participant WS as WebSocket Server
    participant F2 as Frontend 2
    participant U2 as User 2
    participant DB as Database

    U1->>F1: Type message
    F1->>WS: Send message via WebSocket
    WS->>DB: Save message to database
    WS->>F2: Broadcast message to User 2
    F2->>U2: Display message
    U2->>F2: Type reply
    F2->>WS: Send reply via WebSocket
    WS->>DB: Save reply to database
    WS->>F1: Broadcast reply to User 1
    F1->>U1: Display reply
```

Exam Management Flow

```
flowchart TD
    A[Teacher Creates Exam] --> B[Set Exam Details]
    B --> C[Add Questions]
    C --> D[Set Time Limit]
    D --> E[Publish Exam]
    E --> F[Students See Available Exams]
    F --> G[Student Starts Exam]
    G --> H[Timer Starts]
    H --> I[Student Answers Questions]
    I --> J{Time Up?}
    J -->|No| I
    J -->|Yes| K[Auto Submit]
    I --> L[Manual Submit]
    K --> M[Grade Exam]
    L --> M
    M --> N[Calculate Score]
    N --> O[Save Results]
    O --> P[Notify Student]
    P --> Q[Update Grade Report]
```

File Upload Flow

```
sequenceDiagram
    participant U as User
    participant F as Frontend
    participant C as FileController
    participant S as FileService
    participant FS as FileStorage
    participant DB as Database
```

```
    U->>F: Select file to upload
    F->>C: POST /api/files/upload
    C->>S: processFileUpload(file)
    S->>FS: Save file to storage
    FS-->>S: File URL
    S->>DB: Save file metadata
    DB-->>S: File record
    S-->>C: File upload response
    C-->>F: 200 OK + file URL
    F-->>U: Show upload success
```

Notification System Flow

```
flowchart TD
    A[System Event] --> B[Notification Service]
    B --> C{Event Type}

    C -->|Assignment Created| D[Notify Students]
    C -->|Grade Posted| E[Notify Student & Parent]
    C -->|Leave Request| F[Notify Teacher/Parent]
    C -->|Exam Scheduled| G[Notify Students]
    C -->|Chat Message| H[Notify Recipient]

    D --> I[Send Email]
    E --> I
    F --> I
    G --> I
    H --> J[Real-time Notification]

    I --> K[Update Notification Table]
    J --> K
    K --> L[Display in UI]
```

Database Schema Relationships

```
erDiagram
    USERS ||--o{ STUDENTS : has
    USERS ||--o{ TEACHERS : has
    USERS ||--o{ PARENTS : has

    STUDENTS ||--o{ ATTENDANCE : tracks
    STUDENTS ||--o{ GRADES : receives
    STUDENTS ||--o{ ASSIGNMENT_SUBMISSIONS : submits
    STUDENTS ||--o{ LEAVE_REQUESTS : requests
    STUDENTS ||--o{ EXAM_ATTEMPTS : takes

    TEACHERS ||--o{ ATTENDANCE : marks
    TEACHERS ||--o{ GRADES : assigns
```

```
    TEACHERS ||--o{ ASSIGNMENTS : creates
    TEACHERS ||--o{ LEAVE_REQUESTS : approves
    TEACHERS ||--o{ EXAMS : creates

    ASSIGNMENTS ||--o{ ASSIGNMENT_SUBMISSIONS : receives
    EXAMS ||--o{ EXAM_QUESTIONS : contains
    EXAMS ||--o{ EXAM_ATTEMPTS : has

    CHAT_ROOMS ||--o{ CHAT_MESSAGES : contains
    USERS ||--o{ CHAT_MESSAGES : sends

    PARENTS ||--o{ STUDENTS : monitors
    PARENTS ||--o{ LEAVE_REQUESTS : approves
```

Security Flow

```
flowchart TD
    A[Request] --> B{JWT Token Present?}
    B -->|No| C[Redirect to Login]
    B -->|Yes| D[Validate JWT Token]
    D --> E{Token Valid?}
    E -->|No| C
    E -->|Yes| F[Check User Role]
    F --> G{Has Permission?}
    G -->|No| H[Return 403 Forbidden]
    G -->|Yes| I[Process Request]
    I --> J[Return Response]

    C --> K[User Login]
    K --> L[Generate New JWT]
    L --> M[Store Token]
    M --> N[Redirect to Original Request]
```

System Deployment Architecture

```
graph TB
    subgraph "Client Layer"
        Browser[Web Browser]
        Mobile[Mobile Browser]
    end

    subgraph "Load Balancer"
        LB[NGINX Load Balancer]
    end

    subgraph "Frontend Layer"
        React1[React App Instance 1]
        React2[React App Instance 2]
```

```
    end

    subgraph "Backend Layer"
        Spring1[Spring Boot Instance 1]
        Spring2[Spring Boot Instance 2]
    end

    subgraph "Database Layer"
        MySQL[(MySQL Database)]
        Redis[(Redis Cache)]
    end

    subgraph "External Services"
        Email[Email Service]
        Storage[File Storage]
        CDN[CDN]
    end

    Browser --> LB
    Mobile --> LB
    LB --> React1
    LB --> React2
    React1 --> Spring1
    React1 --> Spring2
    React2 --> Spring1
    React2 --> Spring2
    Spring1 --> MySQL
    Spring2 --> MySQL
    Spring1 --> Redis
    Spring2 --> Redis
    Spring1 --> Email
    Spring2 --> Storage
    React1 --> CDN
    React2 --> CDN
```

Error Handling Flow

```
flowchart TD
    A[Request] --> B{Valid Request?}
    B -->|No| C[Return 400 Bad Request]
    B -->|Yes| D[Process Request]
    D --> E{Database Error?}
    E -->|Yes| F[Return 500 Internal Server Error]
    E -->|No| G{Authentication Error?}
    G -->|Yes| H[Return 401 Unauthorized]
    G -->|No| I{Authorization Error?}
    I -->|Yes| J[Return 403 Forbidden]
    I -->|No| K[Return Success Response]

    C --> L[Log Error]
```

```
    F --> L
    H --> L
    J --> L
    K --> M[Log Success]
```

This comprehensive flow chart demonstrates the complete architecture and data flow of the ePathshala system, covering authentication, user interactions, real-time features, and system deployment.