SplitSmart

Sprint 1

Team:
Mohammad Areeb
Prabhkrit Singh
Samarjeet Singh

Table of Contents

CRC Cards:

User	
Responsibilities: Store user email, password hash, name, join date; Validate email format; Manage authentication state	Collaborators: AuthController, Group, UserRepository

AuthController	
Responsibilities: Handle registration requests; Handle login requests; Generate JWT tokens; Validate credentials	Collaborators: User, UserRepository, PasswordUtil, JWTUtil

UserRepository	
Responsibilities: Save user to database; Retrieve user by email; Retrieve user by ID; Check email uniqueness; Fetch user profile	Collaborators: User, MongoDB

PasswordUtil	
Responsibilities: Hash passwords with bcrypt; Compare password with hash; Validate password strength	Collaborators: User

JWTUtil	
Responsibilities: Generate JWT token; Sign token with secret; Verify token; Extract user ID from token	Collaborators: AuthController

Group	
Responsibilities: Store group name, description, members; Track group admin; Store creation date; Manage member list	Collaborators: User, GroupController, GroupRepository

GroupController	
Responsibilities: Handle create group requests; Handle list groups requests; Validate group data; Set admin on creation	Collaborators: Group, GroupRepository, User

GroupRepository	
Responsibilities: Save group to database; Retrieve groups by user; Retrieve group by ID; Add member to group; Get member count	Collaborators: Group, MongoDB

AuthMiddleware	
Responsibilities: Check JWT token validity; Extract user from token; Protect authenticated routes; Return 401 on invalid token	Collaborators: JWTUtil, User

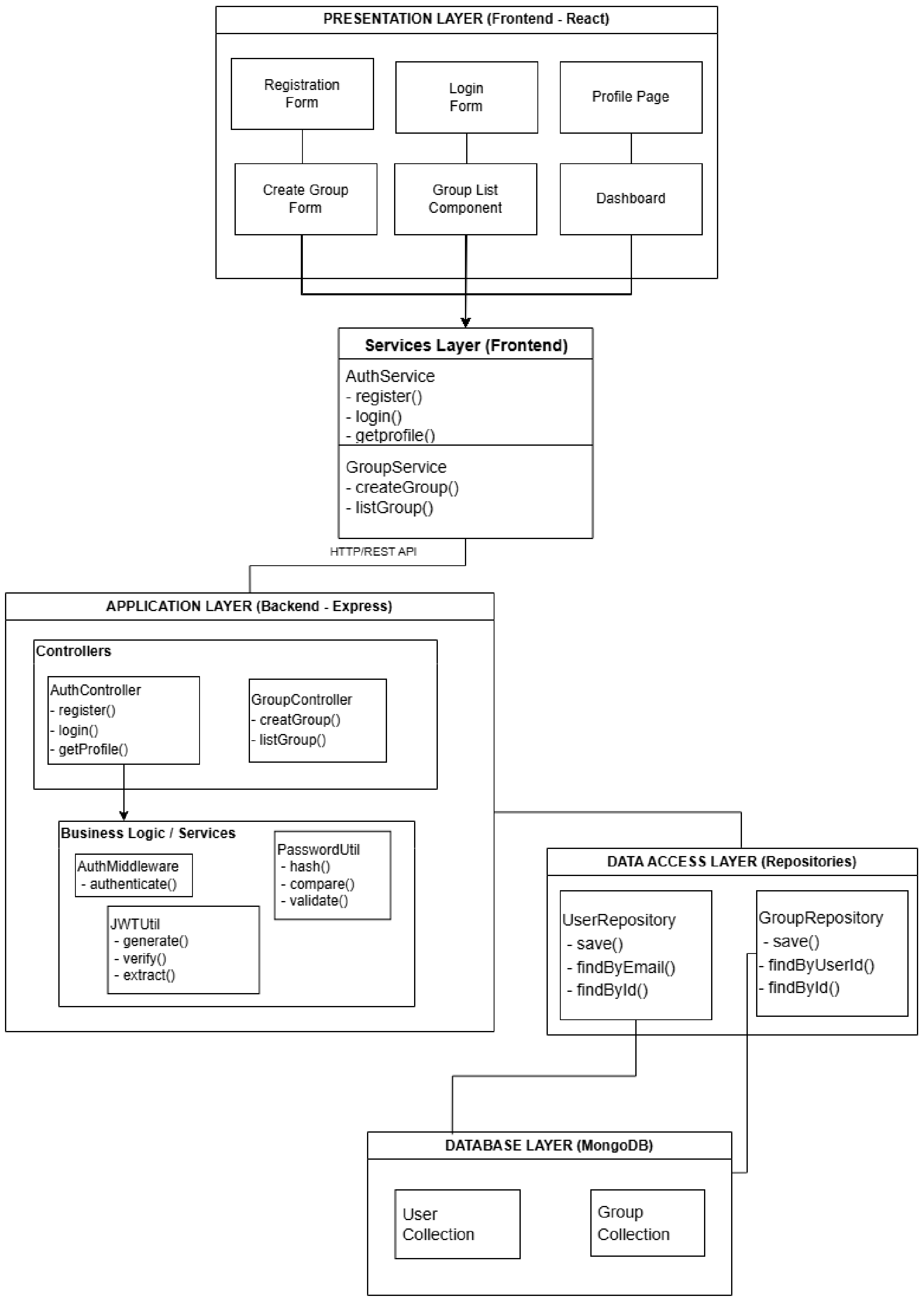
RegistrationForm (Frontend)	
Responsibilities: Render email, password, confirm password fields; Validate form inputs client-side; Provide error feedback; Submit to API	Collaborators: AuthService, User

LoginForm (Frontend)	
Responsibilities: Render email and password fields; Validate inputs; Submit credentials to API; Store token on success	Collaborators: AuthService, User, TokenStorage

GroupForm (Frontend)	
Responsibilities: Call create group API; Call list groups API; Manage group state; Filter groups by user	Collaborators: GroupForm, GroupList, API

TokenStorage (Frontend)	
Responsibilities: Store JWT in localStorage/sessionStorage; Retrieve token; Clear token on logout; Check token expiry	Collaborators: AuthService

Software Architecture Diagram:



System Interaction with Environment

Dependencies:

- Node.js (v18+)
- MongoDB (v6.0+)
- React (v18+)
- Express.js (v4.18+)
- npm/yarn package manager

External Systems:

- MongoDB Atlas (cloud database)
- JWT signing library
- bcryptjs for password hashing
- CORS for cross-origin requests
- Environment variables for config

Assumptions:

- Users have internet connection
- MongoDB connection always available
- Server runs on localhost:5000, Frontend on localhost:3000
- Modern browsers support ES6+

System Decomposition

Component Breakdown:

Component	Responsibility	Dependencies
AuthController	Handle registration & login requests	UserRepository, PasswordUtil, JWTUtil
GroupController	Handle group creation & listing	GroupRepository, User
UserRepository	CRUD operations for User	MongoDB
GroupRepository	CRUD operations for Group	MongoDB
AuthMiddleware	Protect routes, validate JWT	JWTUtil
PasswordUtil	Secure password handling	bcryptjs
JWTUtil	Token management	jsonwebtoken
Registration/Login Forms	User input capture	AuthService
Groups Dashboard	Display user's groups	GroupService
AuthService (Frontend)	API communication for auth	HTTP Client
GroupService (Frontend)	API communication for groups	HTTP Client

Error Handling Strategy

Backend Errors:

- **400 Bad Request:** Invalid email format, missing fields, weak password
- **409 Conflict:** Email already exists (duplicate registration)
- **401 Unauthorized:** Invalid credentials, expired JWT token
- **404 Not Found:** User or group not found
- **500 Internal Server Error:** Database connection failed, unexpected errors

Frontend Errors:

- **Network Errors:** Display "Connection failed, please try again"
- **Invalid Input:** Show field-level error messages
- **API Errors:** Display user-friendly error toast messages
- **Auth Errors:** Redirect to login on 401, show "Session expired"
- **Form Validation:** Real-time validation with error icons