1. Server Initialization and Setup

graph TD A[Start server.js] --> B{Import Dependencies}; B --> C{Define Configuration JWT_SECRET}; C --> D{Setup SQLite Database}; D -- Run --> E[CREATE TABLE IF NOT EXISTS users]; D -- Run --> F[CREATE TABLE IF NOT EXISTS boards]; E & F --> G{Setup Express App}; G -- Use --> H[express.json]; G -- Use --> I[cors]; G -- Use --> J[cookieParser]; G --> K{Define Auth Utilities generateTokens, verifyToken, etc. }; K --> L{Define Auth Middleware authenticateToken }; L --> M{Define Express Routes /register, /login, /boards, /pin, /reply }; M --> N{Create HTTP Server from Express App}; N --> O{Setup Socket.IO on HTTP Server}; O -- Use --> P[Socket.IO Auth Middleware]; O -- On 'connection' --> Q{Define Socket Event Handlers pinBoard, addReply }; Q --> R{Start HTTP Server Listen on Port 3000 }; R --> S[Log Server running...]; S --> T[End Setup / Server Running];
**2. HTTP Request Flow Example: POST /pin **

graph TD A[Client sends POST /pin request w/ data & accessToken cookie] --> B{Express Receives Request}; B --> C{Middleware: express.json }; C --> D{Middleware: cors }; D --> E{Middleware: cookieParser }; E --> F{Middleware: authenticateToken}; F --> G{Token Found?}; G -- No --> H[Send 401 Unauthorized]; G -- Yes --> I{Verify Token}; I -- Invalid --> J[Send 403 Forbidden]; I -- Valid --> K{Attach user data to req.user}; K --> L{Route Handler: /pin}; L --> M{Input Data Valid? text, location }; M -- No --> N[Send 400 Bad Request]; M -- Yes --> O{DB: INSERT INTO boards user_id from req.user, text, location }; O -- Error --> P[Send 500 Server Error]; O -- Success --> Q[Send 200 OK Board pinned...]; H & J & N & P & Q --> R[End Request];
**3. WebSocket Connection and Event Flow Example: 'pinBoard' event **

graph TD subgraph Connection Phase A[Client attempts WebSocket connection w/ accessToken cookie] --> B{Server Receives Handshake}; B --> C{Socket.IO Auth Middleware}; C --> D{Parse accessToken from Cookies}; D --> E{Token Found?}; E -- No --> F[Reject Connection Error: No Token]; E -- Yes --> G{Verify Token}; G -- Invalid --> H[Reject Connection Error: Invalid Token]; G -- Valid --> I{Attach user data to socket.user}; I --> J[Connection Established]; J --> K{Trigger 'connection' event on server}; end subgraph Event Handling 'pinBoard' L[Client emits 'pinBoard' event w/ data text, location] --> M{Server Receives 'pinBoard' on

Socket}; M --> N{Event Handler: 'pinBoard'}; N --> O{socket.user exists?}; O -- No --> P[Emit 'error' to Client]; O -- Yes --> Q{Input Data Valid? text, location }; Q -- No --> R[Log Error / Do Nothing]; Q -- Yes --> S{DB: INSERT INTO boards user_id from socket.user, text, location }; S -- Error --> T[Log DB Error]; S -- Success --> U{Get new board ID}; U --> V{DB: SELECT new board data + username JOIN users}; V -- Error --> W[Log DB Error]; V -- Success --> X{Parse replies JSON}; X --> Y[io.emit 'boardUpdate', { action: 'add', data: board }]; end K --> L; F & H & P & R & T & W & Y --> Z[End Event/Connection];
**4. User Authentication Flow Example: POST /login **

graph TD A[Client sends POST /login request w/ email & password] --> B{Express Receives Request}; B --> C{Middleware Chain JSON, CORS, CookieParser }; C --> D{Route Handler: /login}; D --> E{Email & Password Provided?}; E -- No --> F[Send 400 Bad Request]; E -- Yes --> G{Call validateLogin email, password }; G --> H{DB: SELECT * FROM users WHERE email = ?}; H -- Error --> I[Send 500 Server Error]; H -- Not Found --> J[Send 401 Invalid Credentials]; H -- Found --> K{Compare provided password with stored hash bcrypt.compare }; K -- Error --> I; K -- No Match --> J; K -- Match --> L{User Authenticated Get user ID }; L --> M{Generate Access & Refresh Tokens JWT }; M --> N{Set accessToken Cookie httpOnly, secure, short expiry }; N --> O{Set refreshToken Cookie httpOnly, secure, long expiry }; O --> P[Send 200 OK with success message]; F & I & J & P --> Q[End Request];