TECHNICAL DESIGN SPECIFICATION

system.in.idea

OVERVIEW

kord.io
system.in.idea

Kord is a centralized hub for communication over a digital medium, whether that's between friends, businesses, working partners, social groups, etc. Users will be able to chat and communicate in a room with an interactive board. Users will be able to draw on the board or create, manipulate, and interact with resources such as documents, videos, etc. It can be used as a space for saving important links, brainstorming, making timelines, and more.

It is a space where digital interactions have a sense of continuity and connect people to each other, rather than to the ether.

Revision History:
* 0.4 11/14/2014: Design Specification
* 0.3 10/16/2014: Functional Specification
* 0.2 10/02/2014: Proposal
* 0.1 09/15/2014: Profile

Disclaimer:
Any designs and features discussed in this document are subject to change.
UI designs will evolve as we test and feature lists will change as we run into technical problems and find inspiration. Technical details will be discussed in a future document. Note that also, the two wireframe versions of the UI illustrate the different layouts and set ups we will be experimenting with.

TEAM MEMBERS

kord.io
system.in.idea

Julian Kuk: *Project Lead*
Responsible for managing the flow of the team and the distribution of tasks, with a primary focus on front-end development.

James Yanyuk: *Back End Lead*
Primarily focused on back-end development, and general integration of the front-end and database. Manages the application's dedicated server (currently running at http://kord.io).

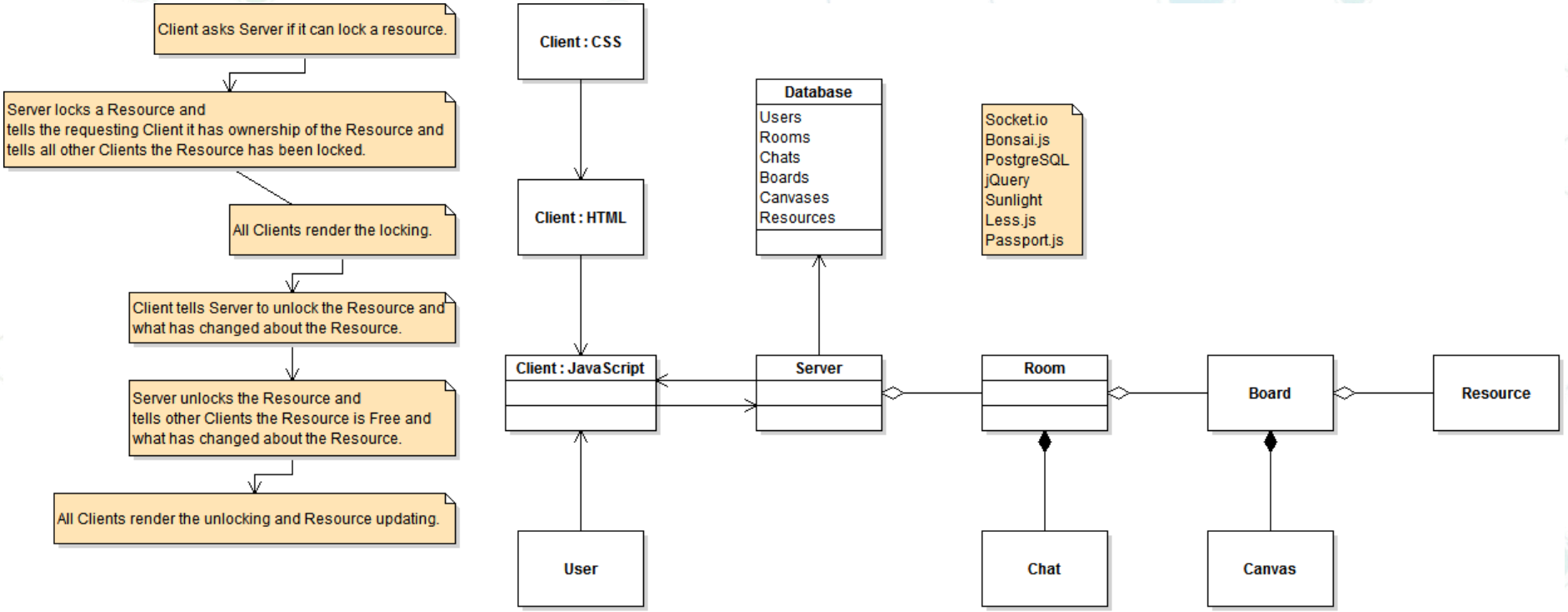
Matthaus Wolff: *Back-End Developer and Marketing Manager*
Primarily focused on back-end development and marketing of the application.

Nam Phan: *Back-End Developer*
Primarily focused on database design and back-end development.

Sara da Silva: *Front-End Lead and Graphic Designer*
Primarily focused on front-end development and graphic design.

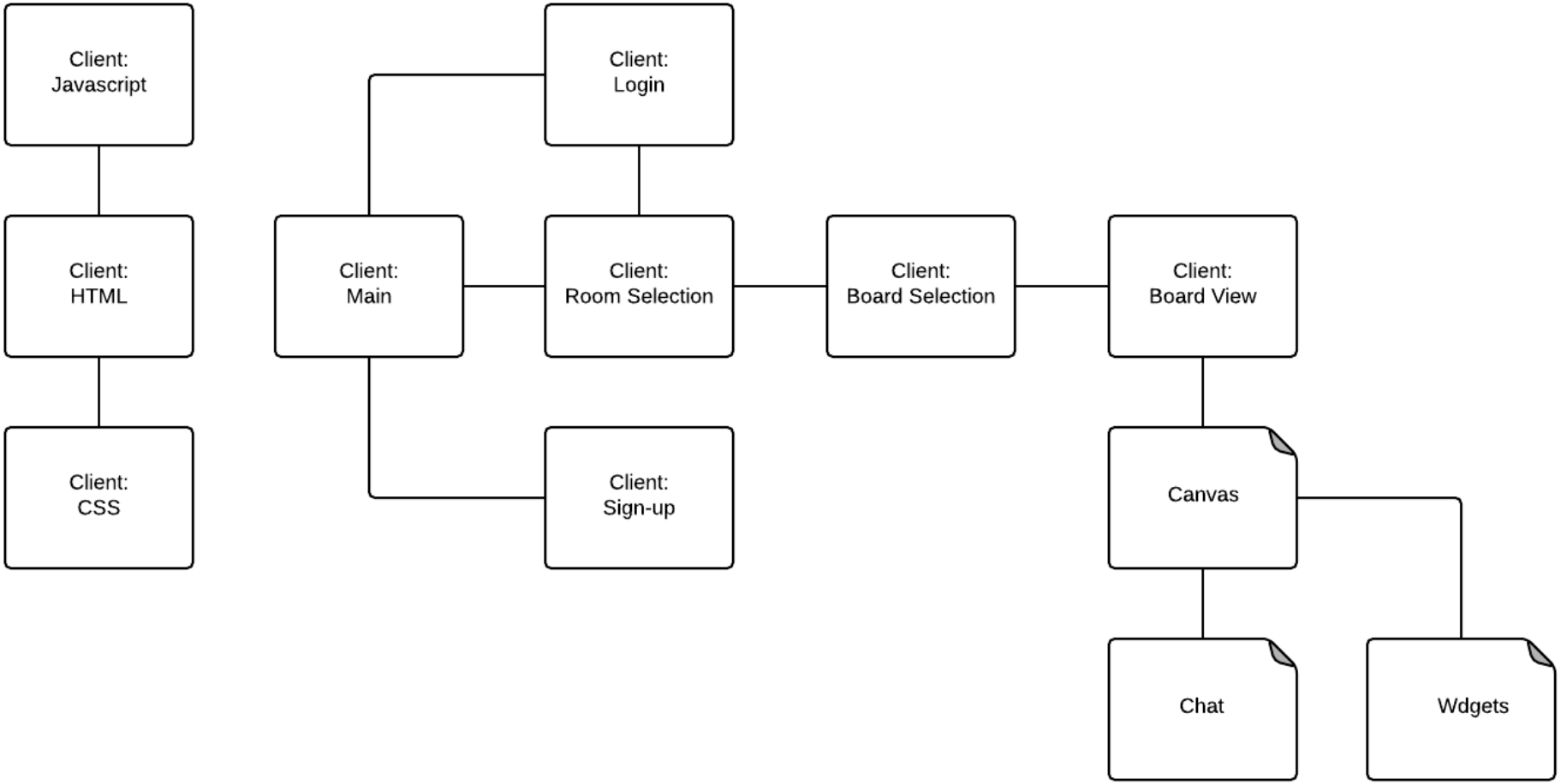
BIRD'S EYE VIEW

kord.io
system.in.idea



CLIENT

kord.io
system.in.idea



CLIENT

kord.io
system.in.idea

Graphics Design
Our application uses HTML5, CSS, and Javascript. It will use the Bootstrap framework in order to develop a responsive and aesthetically pleasing web application.

Main
Users accessing kord.io will be directed to the Main page if they are not logged in. From the main page they will either have the option to sign-up as a new user, or login as an existing users. Users whose sessions has been saved and are currently logged in, will be routed to the Rooms page.

Sign-up
Users who are currently not registered with kord.io will have the chance sign-up using their email address and password. The sign-up process will interact with the user's database and the user class. Upon signing-up a user will be routed to the main page where they will be able to click on "Create a Board" to access our Rooms page.

Login
Users who have previously registered with kord.io will be able to login through this page. A user will login with their email and password, and Passport.js will be used to authenticate their session.

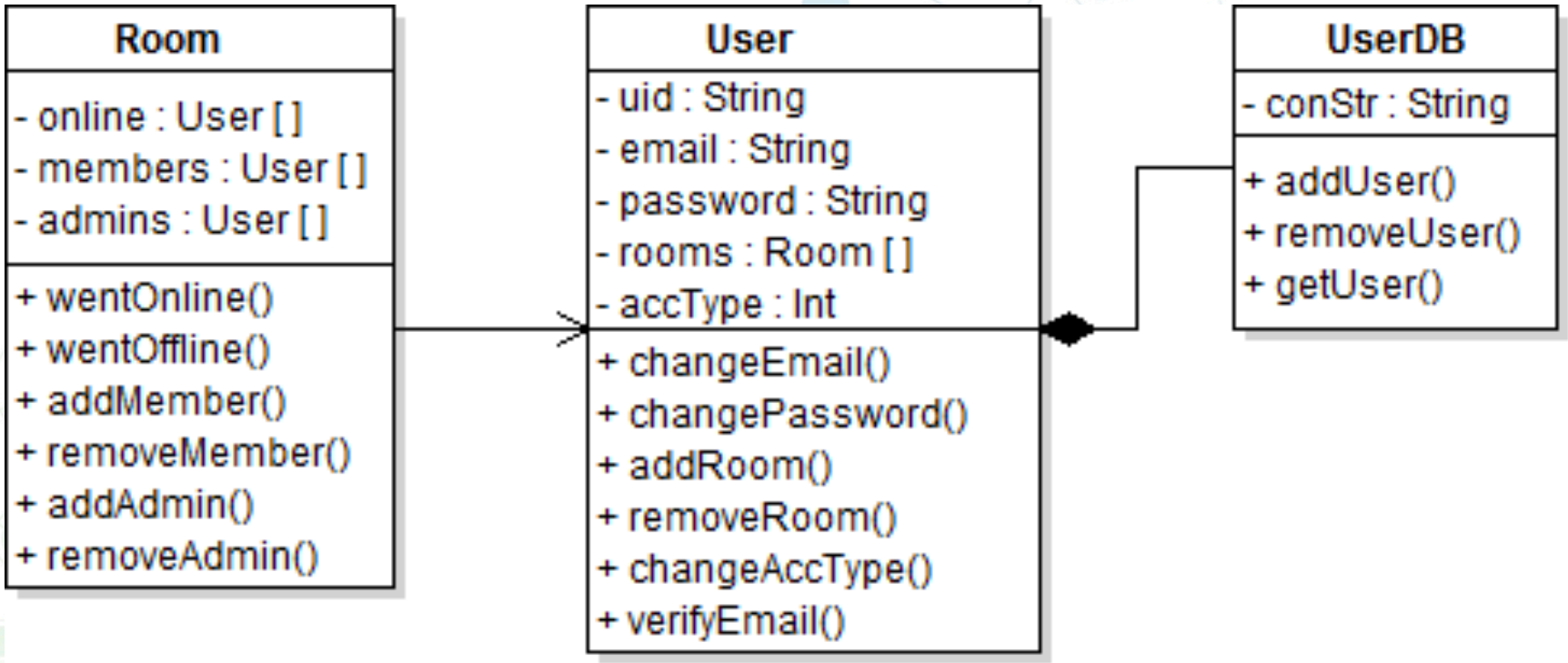
The rooms page, accessed only through proper authentication, will hold a list of rooms a user is currently part of. From this page, users will be able to choose a Room which will direct them to another view containing all of the boards pertaining to this room.

Board Selection
The room page will hold all the boards this room contains. It will also allow a user to add a new board to this room.

Board View
This is where all things come together. A board will consist of a graphics markup, specifically SVG, where all of our components will be placed. While this markup language for graphics will allow our users to draw on the board, users will also be able to layer different components on top as they wish to select from our menu. A board will also have a chat where users currently on the room will be able to communicate with each other, but only one chat is assigned per room. Our board will initially use the BonsaiJS and Socket.io libraries.

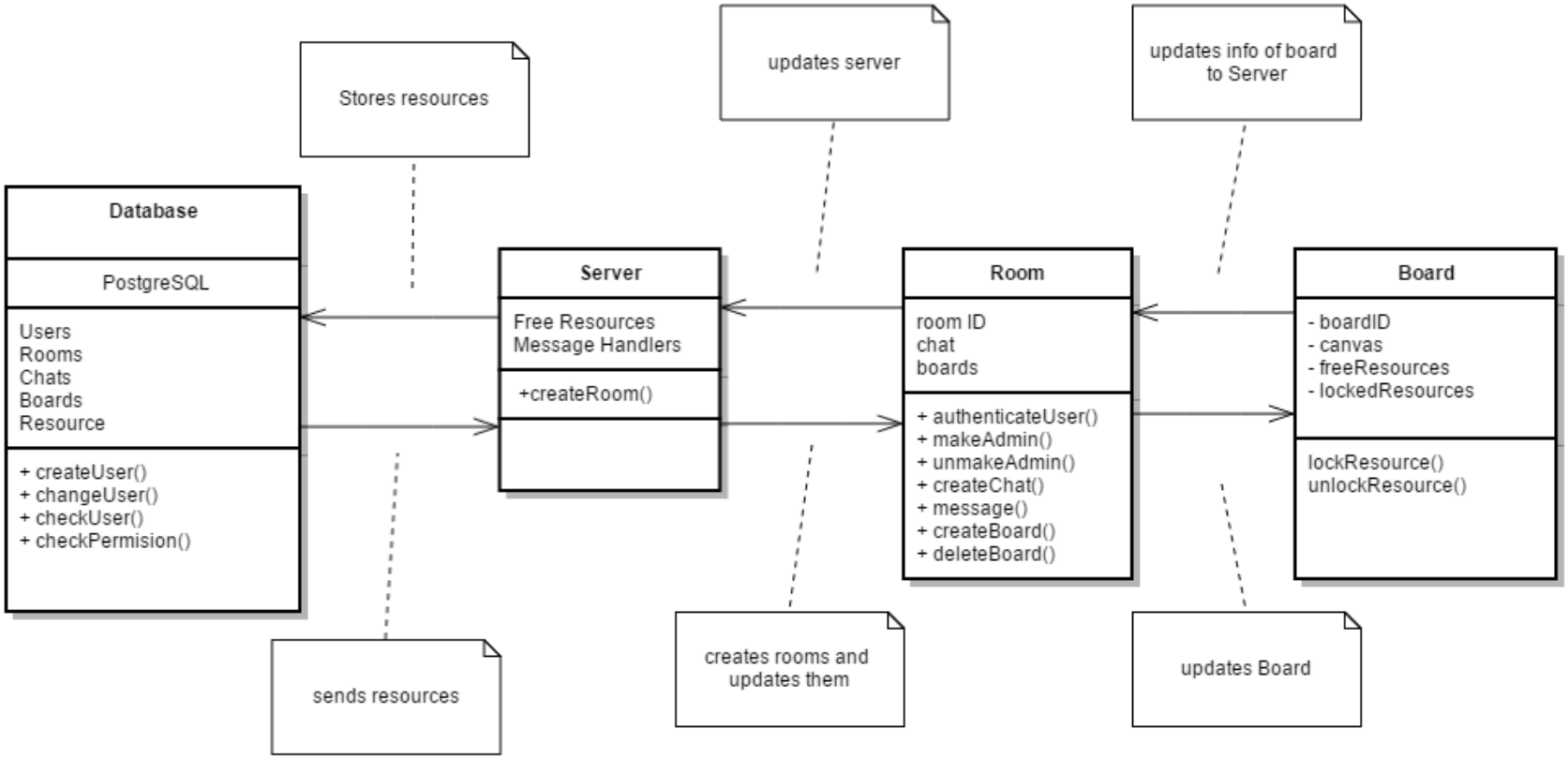
SERVER:

CONNECTION MANAMENT & USER AUTHENTICATION



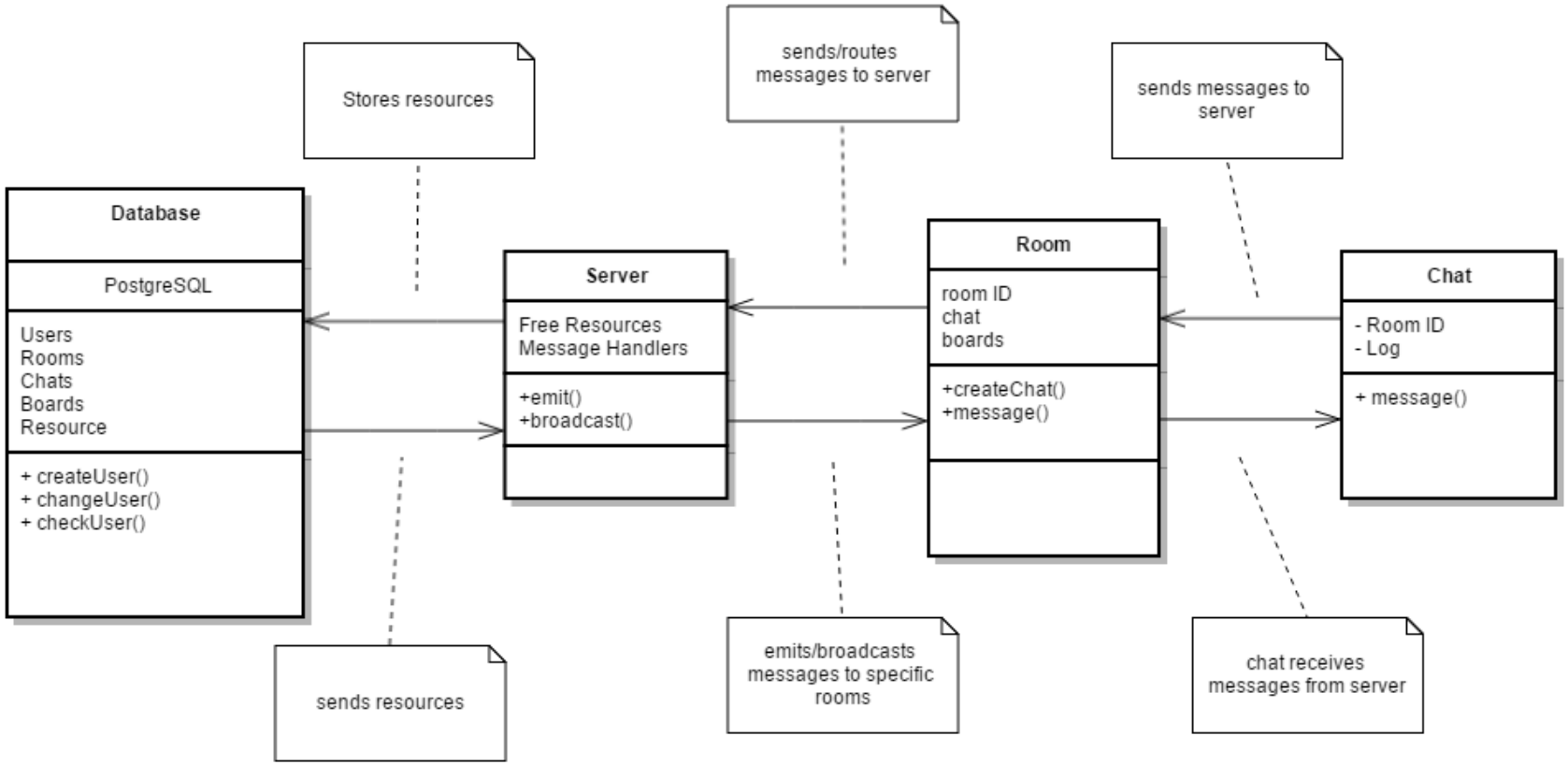
When a user visits the home page, he's presented with the option to login or register for an account. When registered, the user's credentials are stored in the database (password is encrypted), the user's account type is set to unverified, and a verification email is sent to the user. Upon opening the link in the confirmation email, the user's account type is set to verified, and the user is now able to login, bringing them to the user view. From there, they may create a new room, join a room to which they were invited, or enter one they have already joined in the past. From the room or user view, the user may logout, bringing them to the home page.

SERVER:
ROOM AND SERVER INTERACTION



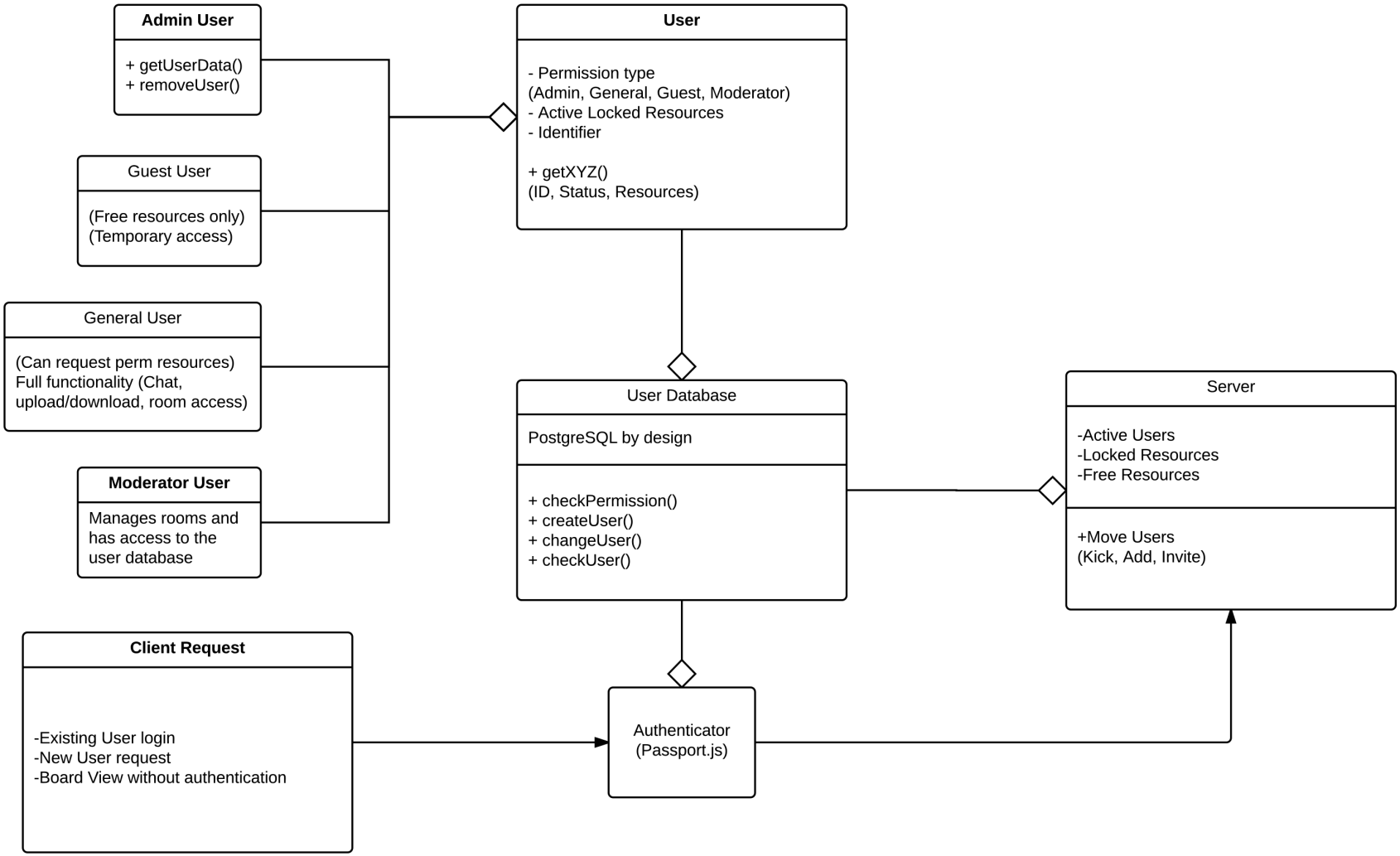
This is the interaction between the rooms and the server. All data will be ultimately stored in the Database which only interacts with the server. The server handles calls from Room and Board. All Rooms have Boards and a single chat. Rooms can assign or unassign admins, authenticate users, create and delete boards. Each room will have a unique ID that they can use to identify themselves from other rooms when interacting with the server and database. Boards have resources that needs to be deemed locked or unlocked for interaction

SERVER:
CHAT AND SERVER INTERACTION



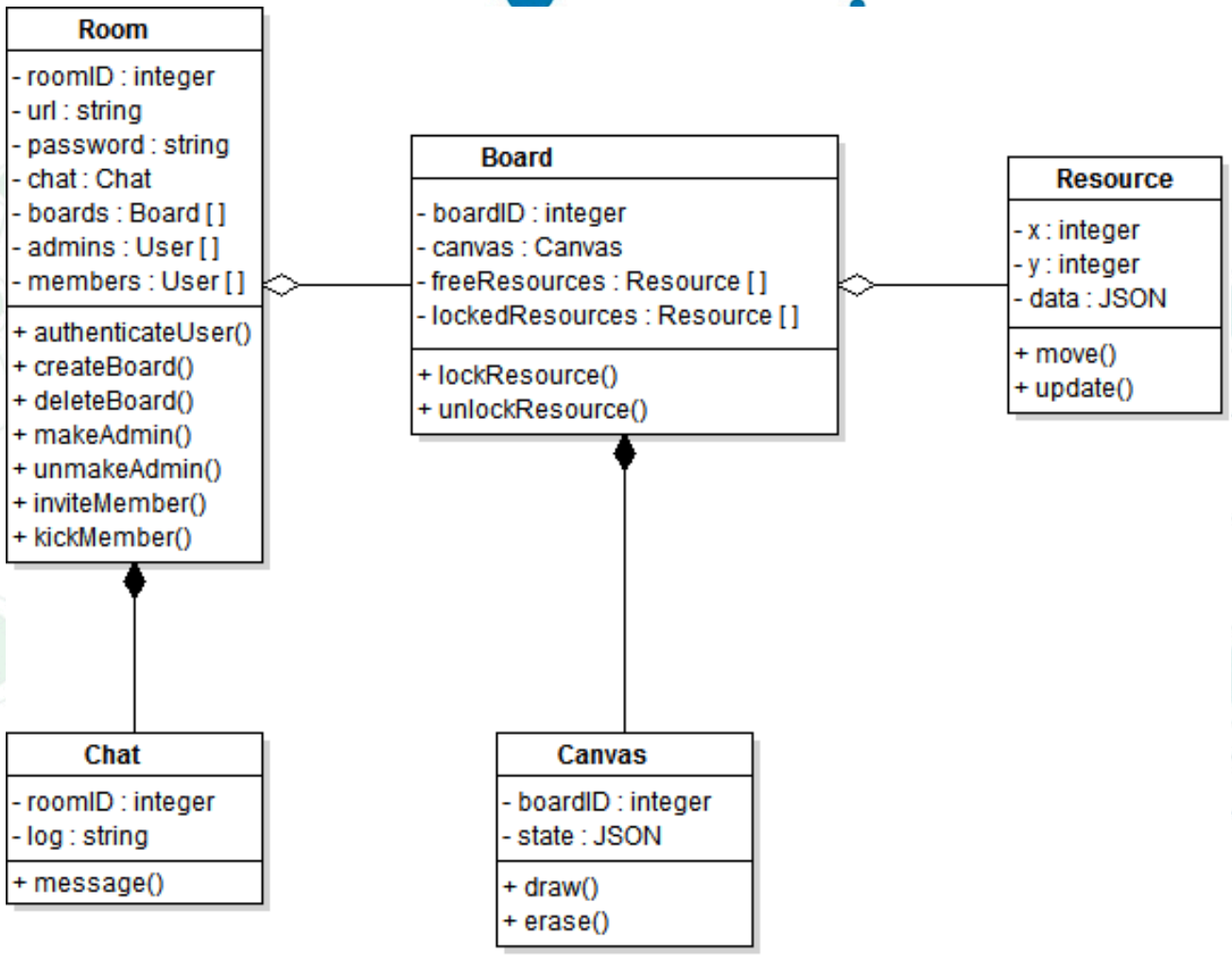
This shows the interaction between the chat and server. The database ultimately stores all resources and only interacts with the server. The server broadcasts or emits messages received from chat to all chats in a room with the room ID passed by the chat call.

DATABASE: USERS



Guest accounts can join open rooms and participate in chat, upload and download files but cannot make permanent rooms. *General users* have the same privileges as guests, but can make permanent rooms. *Moderators* control member privileges within a given room, such as the ability to chat. And a fourth: *admin*, for handling administrative duties throughout the website.

DATABASE: ROOMS



The room is the locus of interaction for users. Here they can chat, draw on the board, or upload resources. The data structures depicted above will be how data is stored. The database will have tables for each data structure, and store the relevant and necessary fields. However, Resources will be stored as serialized objects, because the different resources don't necessarily have the same structure (such as a text file versus an audio file).

EXTERNAL LIBRARIES

kord.io
system.in.idea

Node.js
Serverside JavaScript environment

Express
Web framework for Node.js.

EJS
Javascript template library to seamlessly integrate the Javascript and HTML of our web application.

jQuery
JavaScript library to simplify the client-side scripting of HTML.

Bootstrap
Our web application will use the tools provided by Bootstrap in order to create the front-end design of kord.io.

PostgreSQL
Database infrastructure. We chose this to handle wide-scale development with large amounts of data being handled. Very reliable and stable while being compatible with most platforms.

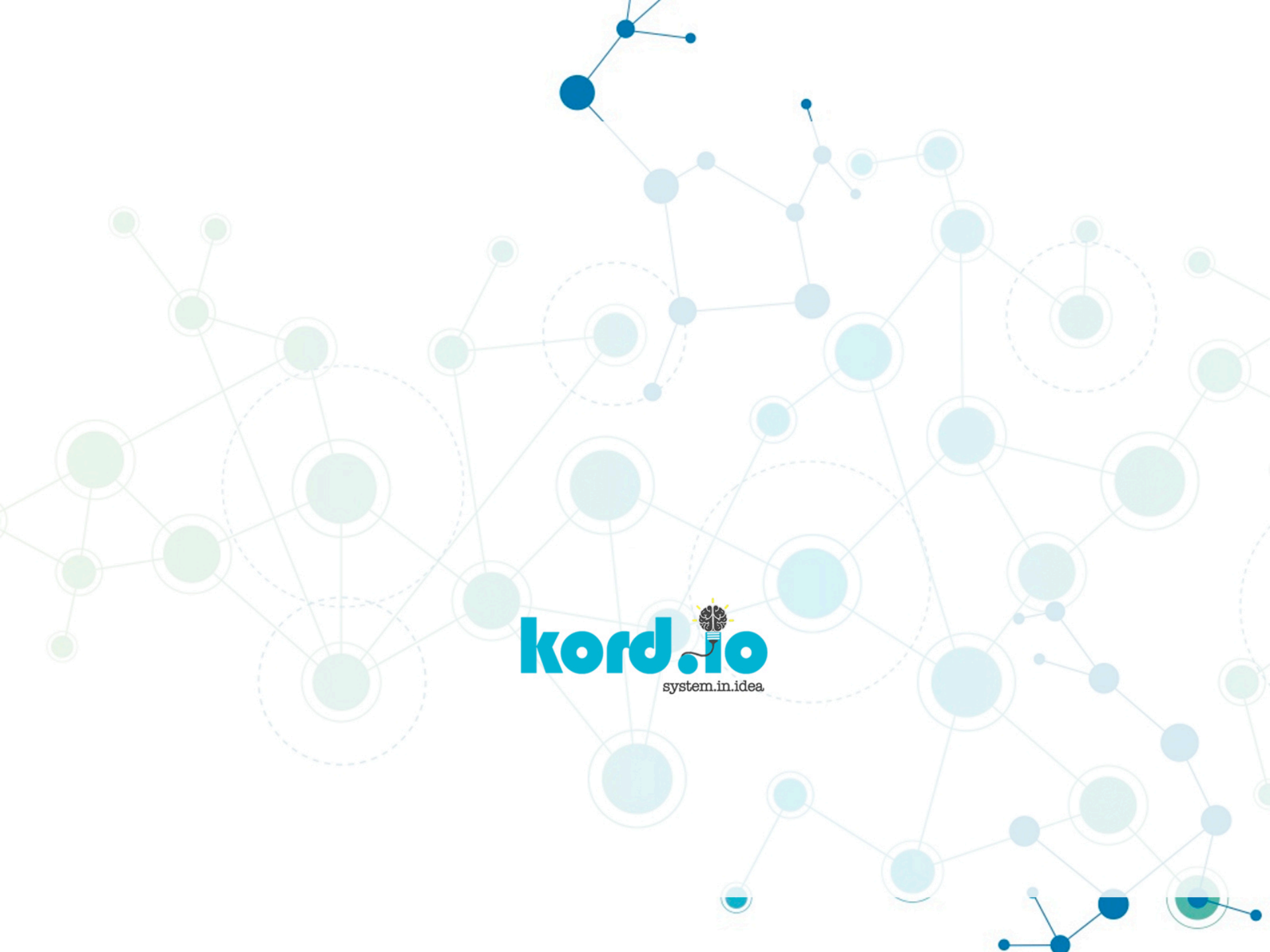
Socket.io
Main engine for the boards. Socket.io simplifies the usage of WebSockets while ensuring compatibility on the users end. This will be the development base for the chat system and routing between rooms.

Bonsai.js
Lightweight graphics library for drawing/animating dynamically on a board. Capable of fairly complicated development and performances while not consuming too many server resources. We prefer SVG over canvas because our app is more likely to have data overflow than it is to require complicated graphics.

Sunlight.js
Used for automatic syntax highlighting within html. Low weight and parallel resources.

Less.js
Client-side CSS pre-processor. Adds more functionality to extend maintainability and creative freedom. Less.js allows for the defining of variables and mix-ins to make full use of Nested syntax. It also has some useful Operational Functions and Mathematical operations that make it more appealing than Sass.

Passport.js
Sweet and simple Node.js authenticator. It supports persistent sessions, and has a dynamic scope. Handles large amounts of users with easy handling of success/failure and has a lightweight code base.


kord.io
system.in.idea