**BOPO**

Web Applications VT18

Group 11

Betina Andersson,

Fressia Merino Espinosa,

Shahad Naji

Supervisor: Joachim Von Hacht

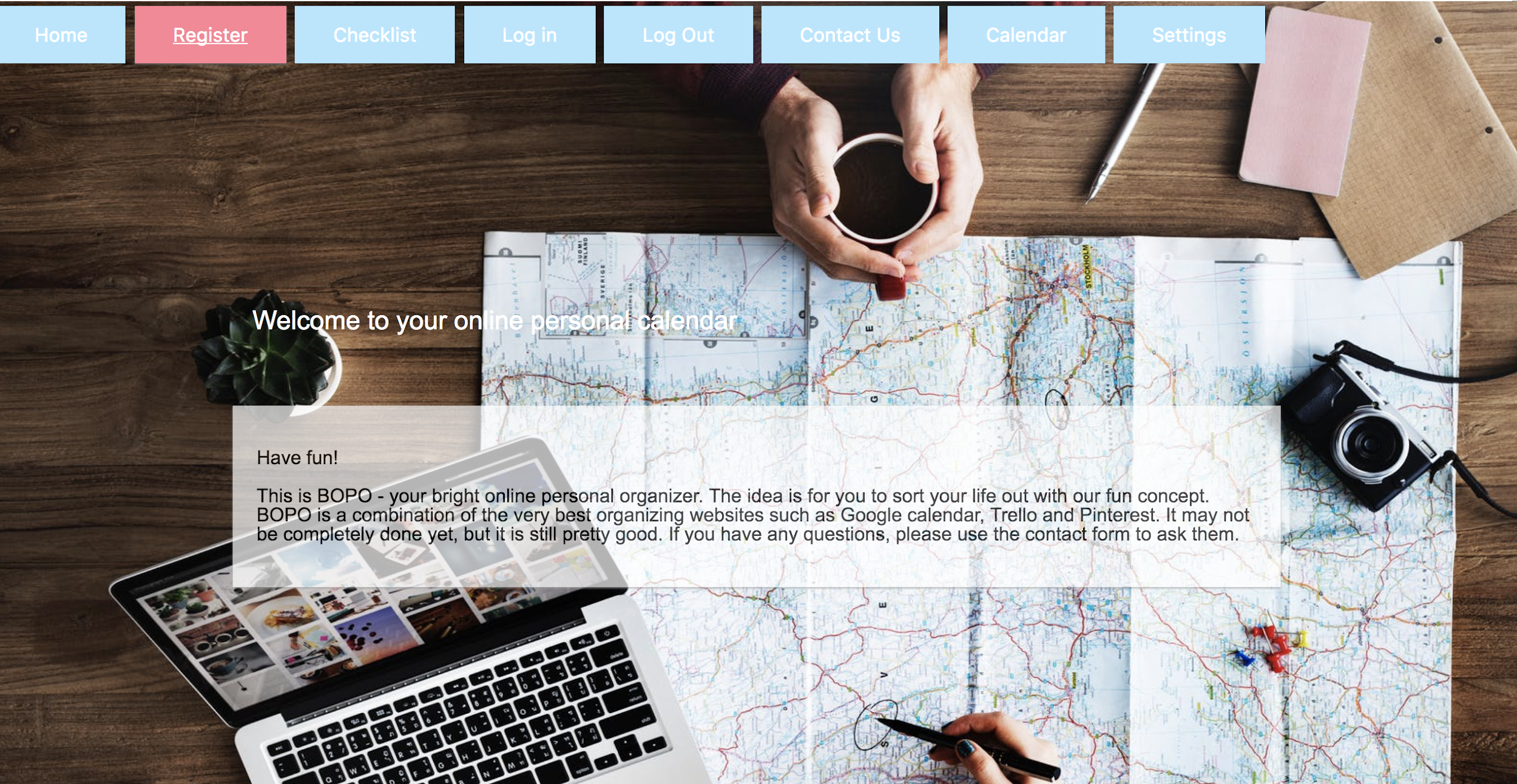
**USE CASES:**

1. Register
2. Log in
3. Log out
4. Add note in checklist
5. Edit note
6. Remove note
7. Search note by category
8. Contact us
9. Upload image
10. Delete account

We decided to make an online personal organizer application with a calendar which we called BOPO - Bright Online Personal Organizer. We used the MVC structure for our application with Node JS environment to develop the web page. With Node JS we could execute the JavaScripts code on server side, and send HTML-pages to the client.

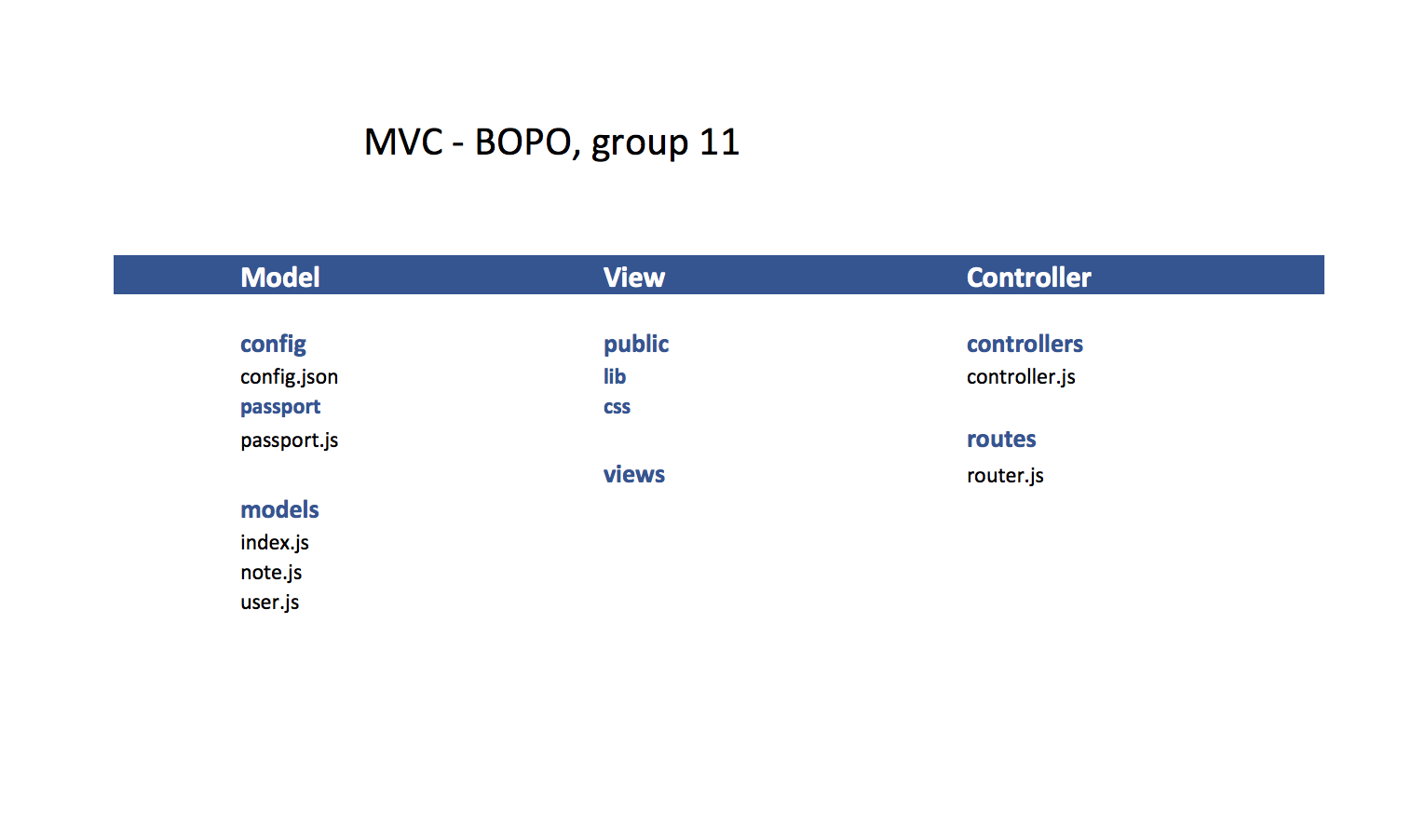
**Example of use case (Log in):**

From the home page, client can click button ”Log In” to be sent over to another html page containing login form. Client should pass in email, and password. With the POST message, the information is sent to the server through routing, the server in its turn searches with Sequelize in the MYSQL database (table users) for matching email and password. Passport is used (with LocalStrategy) to authenticate user’s email and password. Error messages will be shown if email/password would be incorrect and client stays on login page. If passed information is correct, client is directed back to home page and can now visit all other pages.



*The main page of the web site.*

**MVC**

****

The model handles the data and database (MYSQL) connection. For this application, in model you can find *user.js*, *note.js* and *index.js*. In the config folder you also find the connection to the database. We have two tables in our database, one that holds all users, and one that handles the checklist of a user. This structure makes it easy to add more tables to the database, since the connection is already established, and you only need to specify the format of your new table in *models*.

The view handles all HTML-pages which sends information to the client via the *express* framework. There is a view engine in *server.js*.

To handle the connection between server and client side, we use a controller. The controller is in the *routes* folder, and has a lot of GET:s and POST:s. The controller accepts the requests, responses and nexts (req, res, next) and uses them to get information from the server and send back information to client side.

**Short description of files:**

Server.js - Connects to port, starts the application, controls the session

app/config/passport/passport.js - Handles authentication of user, registration of a new user, encrypted passwords and log in.

app/config/config.json - Information about the database.

app/controllers/authcontroller.js - Renders HTML pages.

app/public/css - CSS (design files) for the HTML pages.

app/public/lib - All code for the calendar.

app/routes/router.js - The way between client and server. Handles all the HTTP requests and responses.

app/views - Contains all html pages that are being sent to the client.

*Från vänster: Betina Andersson, Fressia Merino Espinosa, Shahad Naji*