

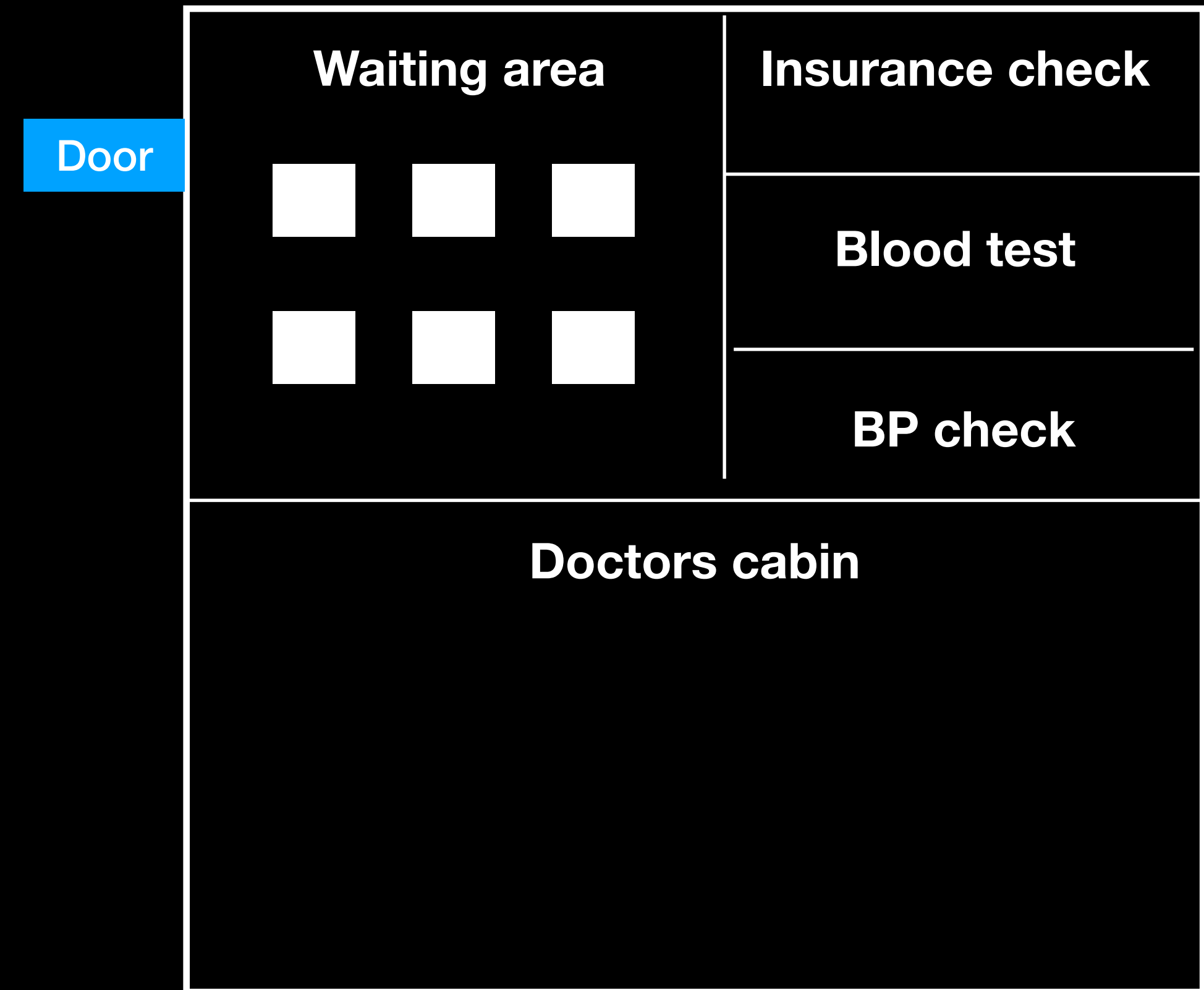
3.1

**Middleware, authentication,
global catches, zod**

Hospital (covid)

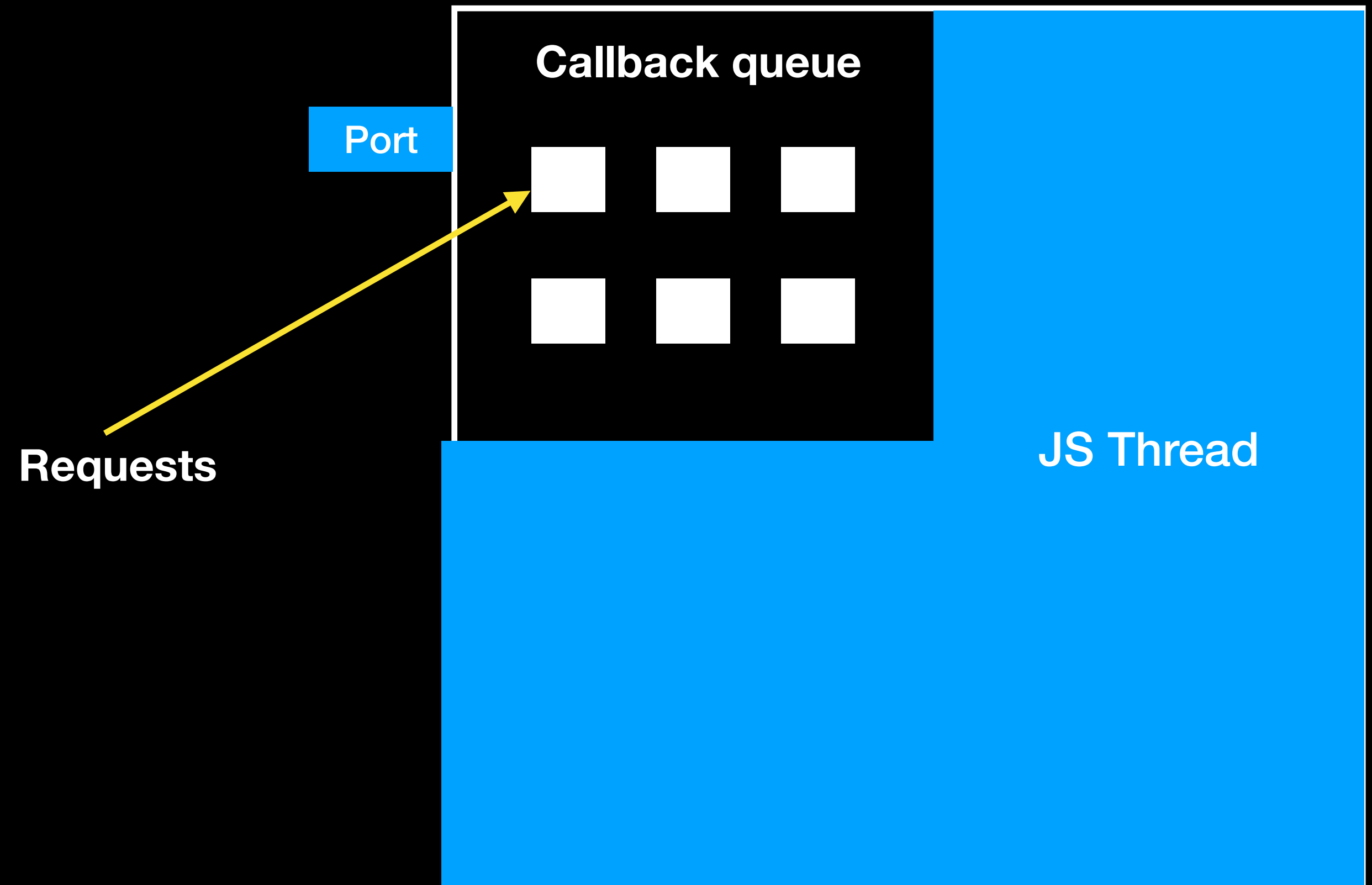
Before you reach the doctor

- 1. Your adhar/insurance info is taken. Only if you have insurance you proceed**
- 2. Blood test is done, only if no STD does use proceed**
- 3. BP is checked, only if BP is reasonable user proceeds**



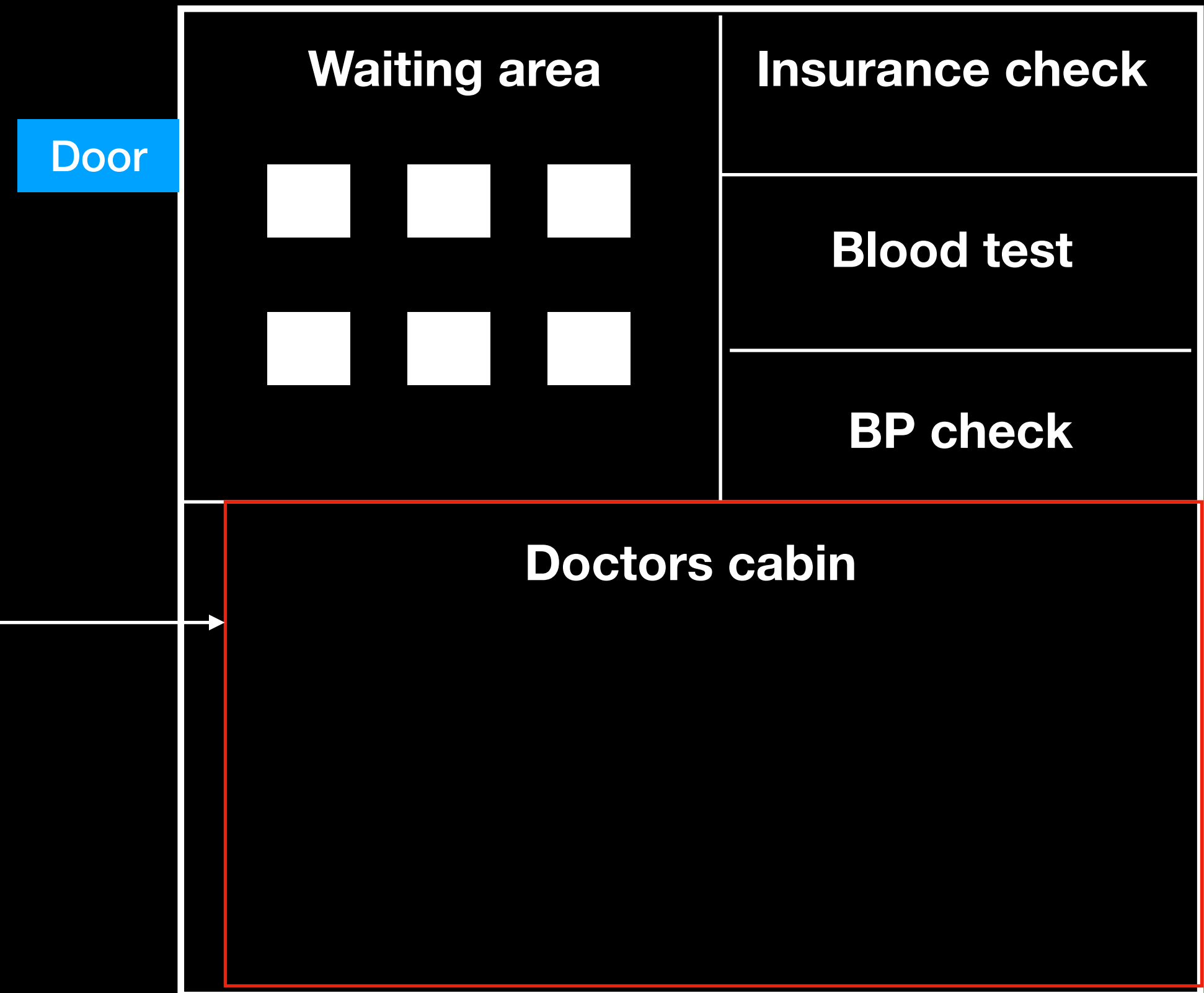
Hospital (covid)

- Before you reach the doctor
1. Your adhar/insurance info is taken
 2. Blood test is done
 3. BP is checked



Equivalent code

```
index.js > ...  
2  const express = require("express");  
3  
4  const app = express();  
5  
6  app.get("/health-checkup", function (req, res) {  
7    // do health checks here  
8    res.send("Your heart is healthy");  
9  });  
10
```



Questions

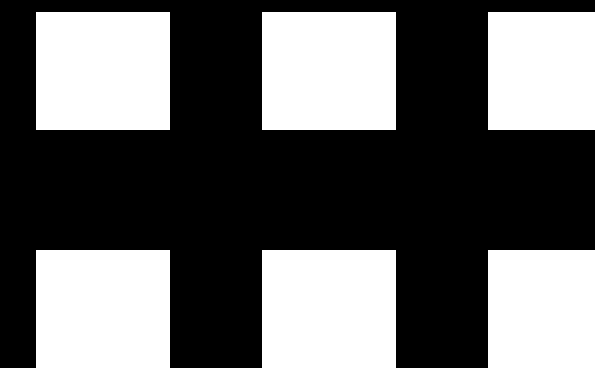
How do you do

1. Auth checks? (Does this user have funds to visit the doctor)
2. Ensure input by the user is valid (BP / blood tests)

```
index.js > ...  
2  const express = require("express");  
3  
4  const app = express();  
5  
6  app.get("/health-checkup", function (req, res) {  
7    // do health checks here  
8    res.send("Your heart is healthy");  
9  });  
10
```

Door

Waiting area



Insurance check

Blood test

BP check

Doctors cabin



Answer - Middlewares

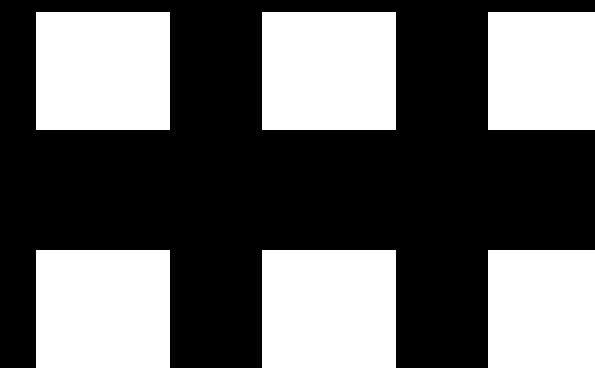
How do you do

1. Auth checks? (Does this user have funds to visit the doctor)
2. Ensure input by the user is valid (BP / blood tests)

```
index.js > ...  
2  const express = require("express");  
3  
4  const app = express();  
5  
6  app.get("/health-checkup", function (req, res) {  
7    // do health checks here  
8    res.send("Your heart is healthy");  
9  });  
10
```

Door

Waiting area



Insurance check

Blood test

BP check

Doctors cabin



Answer - Middlewares

Before we proceed, lets add constraints to our route

1. User needs to send a kidneyId as a query param which should be a number from 1-2 (humans only has 2 kidneys)
2. User should send a username and password in headers

```
index.js > ...  
2  const express = require("express");  
3  
4  const app = express();  
5  
6  app.get("/health-checkup", function (req, res) {  
7    // do health checks here  
8    res.send("Your heart is healthy");  
9  });  
10
```


Answer - Middlewares

Before we proceed, lets add constraints to our route

1. User needs to send a kidneyId as a query param which should be a number from 1-2 (humans only has 2 kidneys)
2. User should send a username and password in headers

```
index.js > ...  
2  const express = require("express");  
3  
4  const app = express();  
5  
6  app.get("/health-checkup", function (req, res) {  
7    // do health checks here  
8    res.send("Your heart is healthy");  
9  });  
10
```


Answer - Middlewares

Before we proceed, lets add constraints to our route

1. User needs to send a kidneyId as a query param which should be a number from 1-2 (humans only has 2 kidneys)
2. User should send a username and password in headers

Username checks

Input validation

```
index.js > f app.get("/health-checkup") callback > ...
2  const express = require("express");
3
4  const app = express();
5
6  app.get("/health-checkup", function (req, res) {
7    // do health checks here
8    const kidneyId = req.query.kidneyId;
9    const username = req.headers.username;
10   const password = req.headers.password;
11
12   if (username !== "harkirat" && password !== "pass") {
13     res.status(403).json({
14       msg: "User doesnt exist",
15     });
16     return;
17   }
18
19   if (kidneyId !== 1 && kidneyId !== 2) {
20     res.status(411).json({
21       msg: "wrong inputs",
22     });
23     return;
24   }
25   // do something with kidney here
26
27   res.send("Your heart is healthy");
28 });
29
```

Answer - Middlewares

Before we proceed, lets add constraints to our route

1. User needs to send a kidneyId as a query param which should be a number from 1-2 (humans only has 2 kidneys)
2. User should send a username and password in headers

What if I tell you to introduce another route that does
Kidney replacement
Inputs need to be the same

Username checks

Input validation

```
index.js > f app.get("/health-checkup") callback > ...
2  const express = require("express");
3
4  const app = express();
5
6  app.get("/health-checkup", function (req, res) {
7    // do health checks here
8    const kidneyId = req.query.kidneyId;
9    const username = req.headers.username;
10   const password = req.headers.password;
11
12   if (username !== "harkirat" && password !== "pass") {
13     res.status(403).json({
14       msg: "User doesnt exist",
15     });
16     return;
17   }
18
19   if (kidneyId !== 1 && kidneyId !== 2) {
20     res.status(411).json({
21       msg: "wrong inputs",
22     });
23     return;
24   }
25   // do something with kidney here
26
27   res.send("Your heart is healthy");
28 });
29
```


Middlewares

What if I tell you to introduce another route that does
Kidney replacement
Inputs need to be the same

Ugly solution - Create a new route, repeat code

```
index.js > f app.put("/replace-kidney") callback > ...

2  const express = require("express");
3
4  const app = express();
5
6  app.get("/health-checkup", function (req, res) {
7    // do health checks here
8    const kidneyId = req.query.kidneyId;
9    const username = req.headers.username;
10   const password = req.headers.password;
11
12   if (username !== "harkirat" && password !== "pass") {
13     res.status(403).json({
14       msg: "User doesnt exist",
15     });
16     return;
17   }
18
19   if (kidneyId !== 1 && kidneyId !== 2) {
20     res.status(411).json({
21       msg: "wrong inputs",
22     });
23     return;
24   }
25   // do something with kidney here
26
27   res.send("Your heart is healthy");
28 });
29
30 app.put("/replace-kidney", function (req, res) {
31   // do health checks here
32   const kidneyId = req.query.kidneyId;
33   const username = req.headers.username;
34   const password = req.headers.password;
35
36   if (username !== "harkirat" && password !== "pass") {
37     res.status(403).json({
38       msg: "User doesnt exist",
39     });
40     return;
41   }
42
43   if (kidneyId !== 1 && kidneyId !== 2) {
44     res.status(411).json({
45       msg: "wrong inputs",
46     });
47     return;
48   }
49   // do kidney replacement logic here
50
51   res.send("Your heart is healthy");
52 });
```

Middlewares

What if I tell you to introduce another route that does
Kidney replacement
Inputs need to be the same

Slightly better solution - Create wrapper fns

```
6 function usernameValidator(username, password) {
7   if (username !== "harkirat" && password !== "pass") {
8     return false;
9   }
10  return true
11 }
12
13 function kidneyValidator(kidneyId) {
14   if (kidneyId !== 1 && kidneyId !== 2) {
15     return false;
16   }
17   return true;
18 }
19
20 app.get("/health-checkup", function (req, res) {
21   // do health checks here
22   const kidneyId = req.query.kidneyId;
23
24   if (!usernameValidator(req.query.username, req.query.password)) {
25     res.status(403).json({
26       msg: "User doesnt exist",
27     });
28     return;
29   }
30
31   if (!kidneyValidator(kidneyId)) {
32     res.status(411).json({
33       msg: "wrong inputs",
34     });
35     return;
36   }
37   // do something with kidney here
38
39   res.send("Your heart is healthy");
40 });
41
42 app.put("/replace-kidney", function (req, res) {
43   // do health checks here
44   const kidneyId = req.query.kidneyId;
45   const username = req.headers.username;
46   const password = req.headers.password;
47
48   if (!usernameValidator(req.query.username, req.query.password)) {
49     res.status(403).json({
50       msg: "User doesnt exist",
51     });
52     return;
53   }
54
55   if (!kidneyValidator(kidneyId)) {
56     res.status(411).json({
57       msg: "wrong inputs",
58     });
59     return;
60   }
61   // do kidney replacement logic here
62
63   res.send("Your heart is healthy");
64 });
```


Middlewares

Defining middleware (just another fn)

```
index.js > f app.get("/heart-check") callback
4  const app = express();
5
6  function userMiddleware(req, res, next) {
7    if (username !== "harkirat" && password !== "pass") {
8      res.status(403).json({
9        msg: "Incorrect inputs",
10      });
11    } else {
12      next();
13    }
14  };
15
16  function kidneyMiddleware(req, res, next) {
17    if (kidneyId !== 1 && kidneyId !== 2) {
18      res.status(403).json({
19        msg: "Incorrect inputs",
20      });
21    } else {
22      next();
23    }
24  };
25
26  app.get("/health-checkup", userMiddleware, kidneyMiddleware, function (req, res) {
27    // do something with kidney here
28
29    res.send("Your heart is healthy");
30  });
31
32
33  app.get("/kidney-check", userMiddleware, kidneyMiddleware, function (req, res) {
34    // do something with kidney here
35
36    res.send("Your heart is healthy");
37  });
38
39  app.get("/heart-check", userMiddleware, function (req, res) {
40    // do something with user here
41
42    res.send("Your heart is healthy");
43  });
44
```

Using the middleware

What if I tell you to introduce another route that does
Kidney replacement
Inputs need to be the same
Best solution - middleware

Last thing in middleware - app.use

```
const app = express();  
app.use(express.json());
```

Other use cases of middleware (assignment) =

- 1. Count the number of requests**
- 2. Find the average time your server is taking to handle requests**

Take a breather
We understand **middlewares**

Why do you need input validation?

Lets see with an example

What if the user sends the wrong body?

```
JS index.js > ...
1  const express = require("express");
2
3  const app = express();
4
5  app.post("/health-checkup", function (req, res) {
6    // do something with kidney here
7    const kidneys = req.body.kidneys;
8    const kidneyLength = kidneys.length;
9
10   res.send("Your kidney length is " + kidneyLength);
11 });
12
13 app.listen(3000);
```

Why do you need input validation?

Lets see with an example

Global catches help you give the user a
Better error message

Error-Handling Middleware: This is a special type of middleware function in Express that has four arguments instead of three (`(err, req, res, next)`). Express recognizes it as an error-handling middleware because of these four arguments.


```
JS index.js > app.post("/health-checkup") callback
1  const express = require("express");
2
3  const app = express();
4
5  app.post("/health-checkup", function (req, res) {
6    // do something with kidney here
7    const kidneys = req.body.kidneys;
8    const kidneyLength = kidneys.length;
9
10   res.send("Your kidney length is " + kidneyLength);
11 });
12
13 app.use((error, req, res, next) => {
14   // console.error(error); // Log the error for debugging
15   res.status(500).send('An internal server error occurred');
16 });
17
18 app.listen(3000);
```

3.1

Middlewares, authentication,
global catches, zod

How can you do better input validation?


This is very hard to scale
What if you expect a complicated input?



```
if (kidneyId != 1 && kidneyId != 2) {  
    return false;  
}
```

How can you do better input validation?

This is where zod comes into the picture



```
if (kidneyId !== 1 && kidneyId !== 2) {  
  return false;  
}
```


How can you do better input validation?

This is where zod comes into the picture

```
JS index.js > [?] z
1  const express = require("express");
2  const z = require("zod");
3  const app = express();
4
5  app.use(express.json())
6
7  const kidneysInput = z.literal("1").or(z.literal("2"));
8
9  app.post("/health-checkup", function (req, res) {
10     // do something with kidney here
11     const kidneyId = req.body.kidneyId;
12     const validation = kidneysInput.safeParse(kidneyId)
13     if (!validation.success) {
14         res.send("Incorrect input");
15         return;
16     }
17     res.send("Your kidney id is " + kidneyId);
18 });
19
20
21 app.listen(3000);
```


How can you do better input validation?

```
import { z } from "zod";

// primitive values
z.string();
z.number();
z.bigint();
z.boolean();
z.date();
z.symbol();

// empty types
z.undefined();
z.null();
z.void(); // accepts undefined

// catch-all types
// allows any value
z.any();
z.unknown();

// never type
// allows no values
z.never();
```

Coercion for primitives

Zod now provides a more convenient way to coerce primitive values.

```
const schema = z.coerce.string();
schema.parse("tuna"); // => "tuna"
schema.parse(12); // => "12"
schema.parse(true); // => "true"
```

3.1

Middlewares, authentication,
global catches, **zod**

Authentication

**As you can tell by now, anyone can send requests to your backend
They can just go to postman and send a request
How do you ensure that this user has access to a certain resource?**

Authentication

Dumb way - Ask user to send username and password in all requests as headers

Authentication

Slightly better way -

- 1. Give the user back a token on signup/signin**
- 2. Ask the user to send back the token in all future requests**
- 3. When the user logs out, ask the user to forget the token (or revoke it from the backend)**

Authentication

Library we need to get comfortable in - **jsonwebtoken**