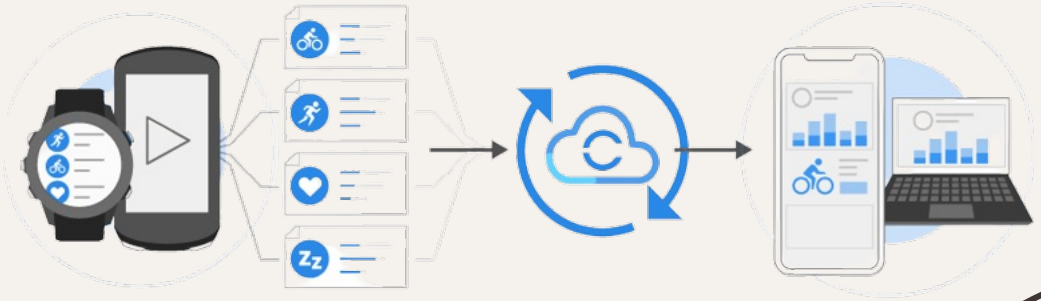
CoachingMate

Garmin API

GA-BoxJelly

Name	Role
Zhidong Zhou	Product Owner
Lingjun Meng	Development Lead
Yuanwei Mao	Developer & Testing Lead
Jialiang Cheng	Scrum Master
Ruiying Feng	Quality Assurance Lead



Outline

01

Background

Zhidong Zhou

02

Technology & Deployment

Yuanwei Mao

03

Product DEMO

Lingjun Meng

04

Fulfillment

Ruiying Feng

05

Handover

Jialiang Cheng

Background

Coaching-Mate

GARMIN Connect

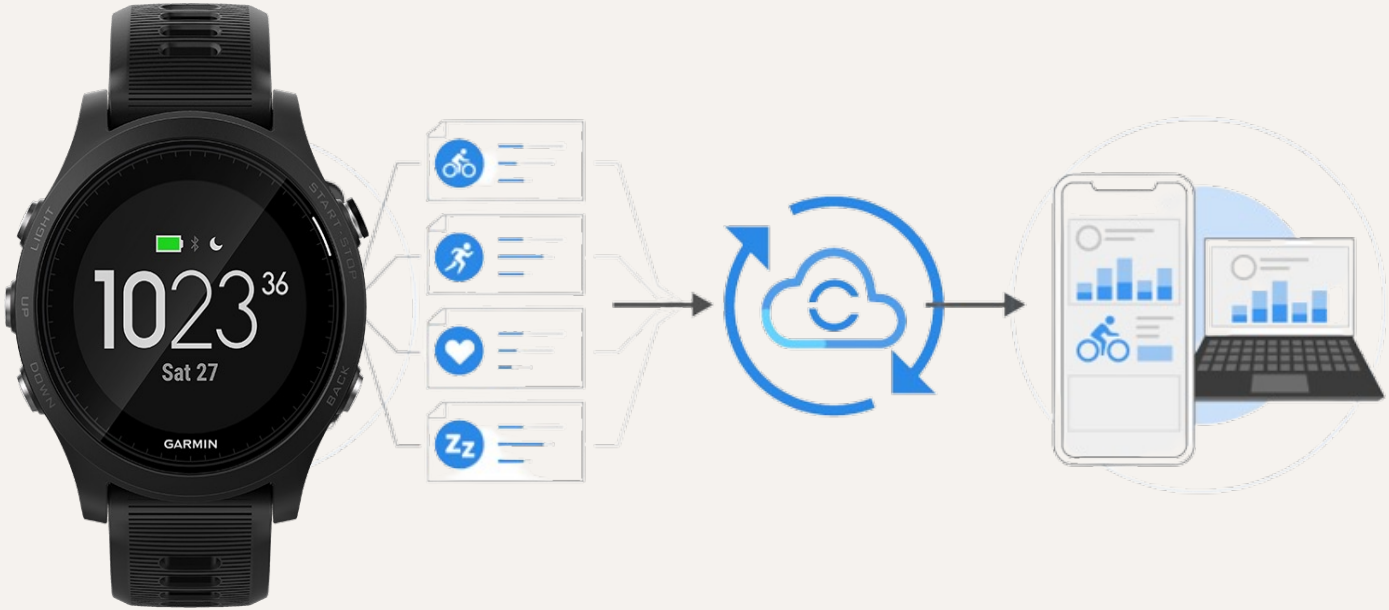
Project Scope

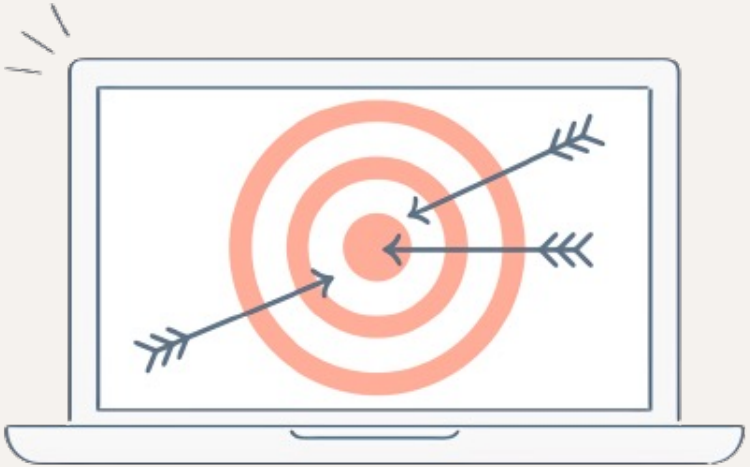
Coaching-Mate

- Coach
- Training Sessions
- Athlete Review
- Web Application

Garmin

- DATA from **Wearable Device**





Project Goal

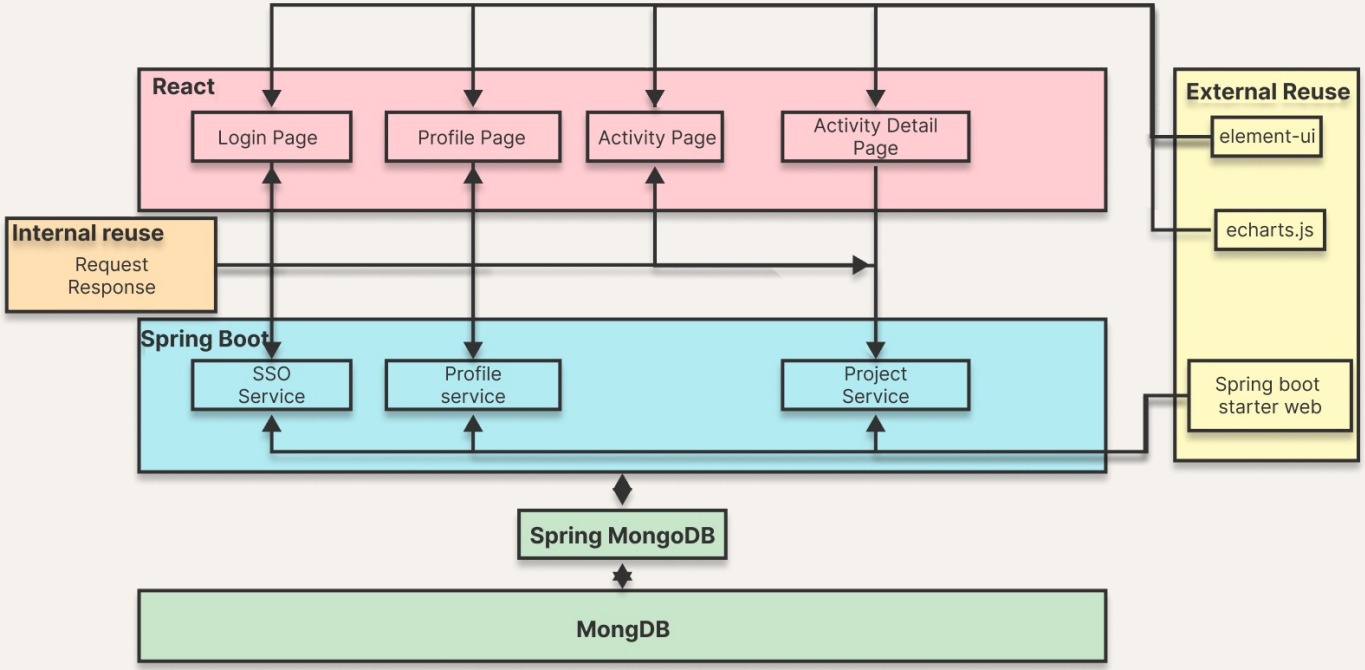
- Data from Garmin Connect
- Activity Data
- Activities Detail
- Based on Previous Code

Technology & Development

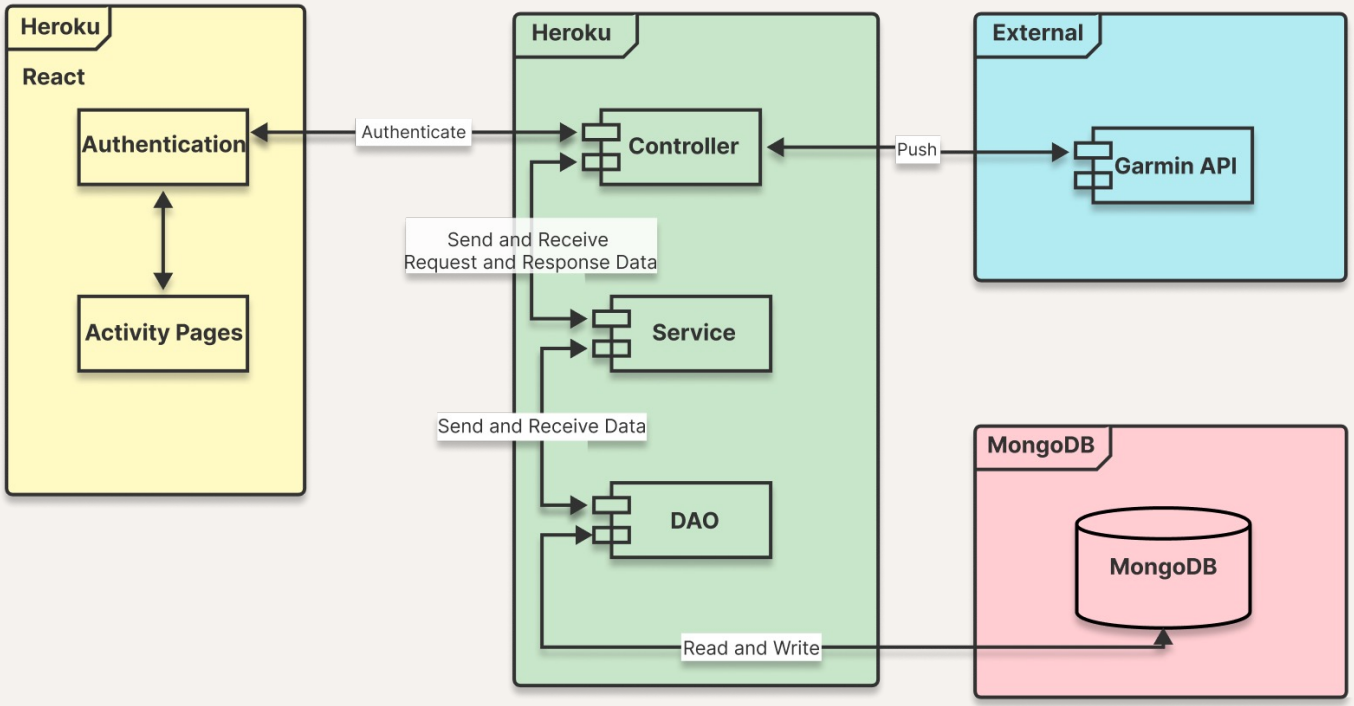
Technology

Deployments

Technology


React
Login Page
Profile Page
Activity Page
Activity Detail Page
External Reuse
element-ui
echarts.js
Internal reuse
Request
Response
Spring Boot
SSO Service
Profile service
Project Service
Spring boot starter web
Spring MongoDB
MongDB

Deployment


Heroku
React
Authentication
Activity Pages
Authenticate
Heroku
Controller
Send and Receive
Request and Response Data
Service
Send and Receive Data
DAO
Read and Write
External
Push
Garmin API
MongoDB
MongoDB


LIVE DEMO

Code Improvement

1. Remove code repetition
2. Add error handle
3. Uniform the responses

Old Request

```
axios({
        method: "post",
        url: `${apiUrl}/login`,
        // url: "http://localhost:8080/login",
        //https://coaching-mate0121.herokuapp.com/login
        headers: {
          "Accept": '*/*',
          "Content-Type": "application/json",
          "Access-Control-Allow-Origin": "*",
          "Access-Control-Allow-Credentials": "true",
        },
        params:{
          username: this.state.username,
          password: this.state.password
        }
      }
    )
    .then((res)=>{
      console.log("response",res)
      this.props.history.push({
        pathname:'/home',
        state: {user: res.data,
              username: res.data.username}
      })
    })
    .catch((error)=> {
      this.setState({isShow : true})
      console.log("error", error)
    })
})
```

New Request

```
const service = axios.create({
  baseURL: process.env.REACT_APP_BASE_API,
});

service.interceptors.request.use(
    (config) => {
      if (store.getState().user.token) {
        config.headers['X-Token'] = getToken()
      }
    return config;
  }
);

service.interceptors.response.use(
  response => {
    const res = response.data
    if (res.statusCode !== 200) {
      // do something
      return Promise.reject('error')
    } else {
      return response.data
    }
  },
  (error) => {// do something
    return Promise.reject(error);
  }
);
```

New Request

```
Old:
axios({
// write the same code again and again.
// even the same api
})


New:
// We just need to create a function for each call.
export function reqUserInfo(data) {
  return request({
    url: '/getUserByToken',
    method: 'post',
    data
  })
}
```

Old Response

In Java controller:
Some Examples:

```
return ResponseEntity.ok(${data});
return new ResponseEntity<>(${data}, HttpStatus.OK);
// Never Handle exceptions!!!
```

New Response

```
public class Response <T> {
    private int statusCode = -1;
    private String message = "Waiting";
    private T data;
}

public class ErrorResponse {
    private Integer statusCode;
    private String message;
    private String exception;
}

public class ResponseHandler implements
ResponseBodyAdvice<Object> {
    // Check if request is success
}
```

New Response

```
public enum ResponseCode {
    SUCCESS(200, "Success"),
    EMAIL_HAS_EXISTED(20001, "Email is already exsited!"),
    PARAM_IS_INVALID(20002, "Param is invalid."),
    USER_IS_NOT_EXISTED(20003, "User is not existed or password is incorrect"),
    USER_HAS_NOT_LOGIN(20004, "User is not login."),
    CHECK_TOKEN_FAIL(20004, "Token verification failed, please log in again.")
}
```

Testing

1. Unit Test
2. Integration Test
3. Acceptance Criteria & Test

Task fulfillment

- In scope
  - Coaching-mate account management
  - Garmin-API connection
  - View Synchronized activity data
- Out scope
  - Website design
  - Analyze the data
- Future demand
  - Retrieve password

Deliverables

Would be delivered

On

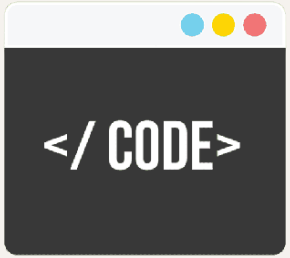
June 13th

Mainly 2 parts:

- Code
- Documents

Code

- Front end
  - React version
- Back end
  - Upgrade version - based on the previous team



Documents

Requirements analysis

- Project Background
- Motivation Model
- Personas
- User stories

Development specification

- Technology
- Deployment

Quality control

- Test cases
- Acceptance criteria
- Acceptance test

User Manual

- Deployment Guideline
- Database structure

Thanks

< Question Time >