캡스톤 디자인

네트워크 시스템 프로젝트

(SKHU's SNS)

최종 보고서

201535015 김홍경

201635017 이재은

201635039 조준서

목차

1 **프로젝트 개요**

2 **개발 환경 및 사용 툴 소개**

1. **개발 환경**
2. React – Native
3. Expo
4. Server
5. 데이터 베이스

3 **시스템 다이어그램**

4 **주요기능**

1. 회원가입 및 로그인 기능
2. 정보게시판
3. 자유게시판
4. 투두리스트
5. 채팅
6. 장터

5 **개발 일정 및 체크리스트**

6 **수정 및 보완점**

- **수정 및 보완점**
- **소감**

1. 프로젝트 개요

현재 성공회대의 자체적인 SNS프로그램이 없다는 점에 착안해 SNS프로그램 제작을 목적으로 하였습니다. 리액트 네이티브를 중점으로 UI를 만들고 ios와 안드로이드 두 개의 플랫폼을 한번에 구현함으로써 접근성을 증대시켰고, 이 하나의 앱을 통해서 학교의 공식 페이스북 계정 페이지의 게시물을 체크하는 동시에 스케줄관리, 학우들간의 소통을 위한 게시판, 중고장터, 채팅 등의 서비스를 이용할 수 있도록 제작하였습니다.

개발환경으로 자바스크립트를 기반으로한 React-native 라이브러리를 주로 사용하였고, 에디터프로그램으로 visual studio code를 이용했으며 expo client를 통해 핸드폰상에서 기능구현의 결과물을 테스트했습니다. 서버로는 node.js와 yarn, 데이터베이스로는 파이어베이스를 사용해 어플리케이션에 데이터를 연동하였습니다.

2. 개발 환경 및 사용 툴 소개

1) 개발 환경

구현 언어 : React-native, javascript, html, css

사용 도구 : expo, visual studio code

server : node.js, yarn

데이터 베이스 : firebase

플랫폼 : ios, android

2) React – Native

React의 접근 방법을 모바일로 확장한 페이스북의 오픈 소스 프로젝트로 ios와 android를 동시에 개발이 가능합니다. react 문법만 익히면, ios의 swift나 android를 각각 익히는 것보다 생산성이 더 높다는 장점을 가집니다. 또한 React 웹으로 확장하기가 용이하며, creative react-native-app과 엑스포를 사용하기 때문에 무거운 개발 툴이 필요 없어 편리하고, 높은 퍼포먼스와 생산성을 가지고 있습니다.

3) Expo

Expo는 ios의 Xcode나 안드로이드 스튜디오 없이 리액트 네이티브로 앱을 제작하는 것을 도와주는 도구입니다. ios와 안드로이드 전용 앱을 동시에 만들 수 있는 장점이 있으며, 핸드폰에 expo 클라이언트를 다운받고 expo에서 제공해주는 qr코드를 핸드폰으로 찍기만 하면 핸드폰에서 개발하고 있는 앱이 동기화가 되기 때문에 앱 테스트가 빠르고 편리합니다.

4) Server

서버로는 node.js와 yarn 두 가지를 사용하였습니다. Node.js는 오픈 소스 서버 환경으로 서버 상의 자바스크립트 런타임 환경을 제공합니다. Node.js에는 npm이라는 기본 패키지 관리자가 있는데 npm은 프로그래머가 node.js 라이브러리의 소스코드를 쉽게 게시하고 공유하게 하며 라이브러리의 설치 업데이트 및 제거를

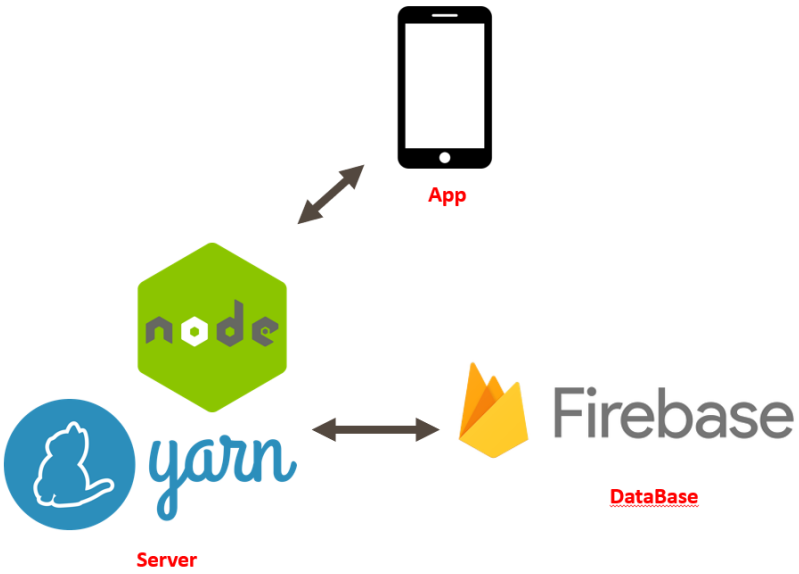
간소화하도록 하는 역할을 하고 있습니다. react-native에서는 npm과 yarn 모두 사용하지만, 저희는 yarn 서버가 자바스크립트 패키지 매니저로서 npm보다 빠르고 안정적이며 보안성이 뛰어나다는 점 때문에 yarn을 선택하게 되었습니다.

5) 데이터베이스

데이터베이스로는 구글에서 제공하는 플랫폼인 파이어베이스를 사용하였습니다. 많은 데이터베이스 중에서 파이어베이스를 사용한 이유는 한정된 기간에 서버를 구축하고 개발하기에 편리하다는 장점 때문이었습니다. 또한 파이어베이스는 네이티브 개발자들이 자주 사용하는 데이터베이스이기도 하고, react-native를 사용할 때 어떤 데이터베이스보다 파이어베이스가 연동성이 뛰어나다고 접했기 때문입니다. 실제로 파이어베이스로 백엔드 개발을 할 때는 서버를 따로 설계, 구현하지 않아도 되기에 파이어베이스는 프론트엔드 개발에 집중할 수 있도록 도와주는 서비스이고, 실시간 데이터베이스, 클라우드 저장소, 호스팅 등 다양한 기능을 제공해줍니다.

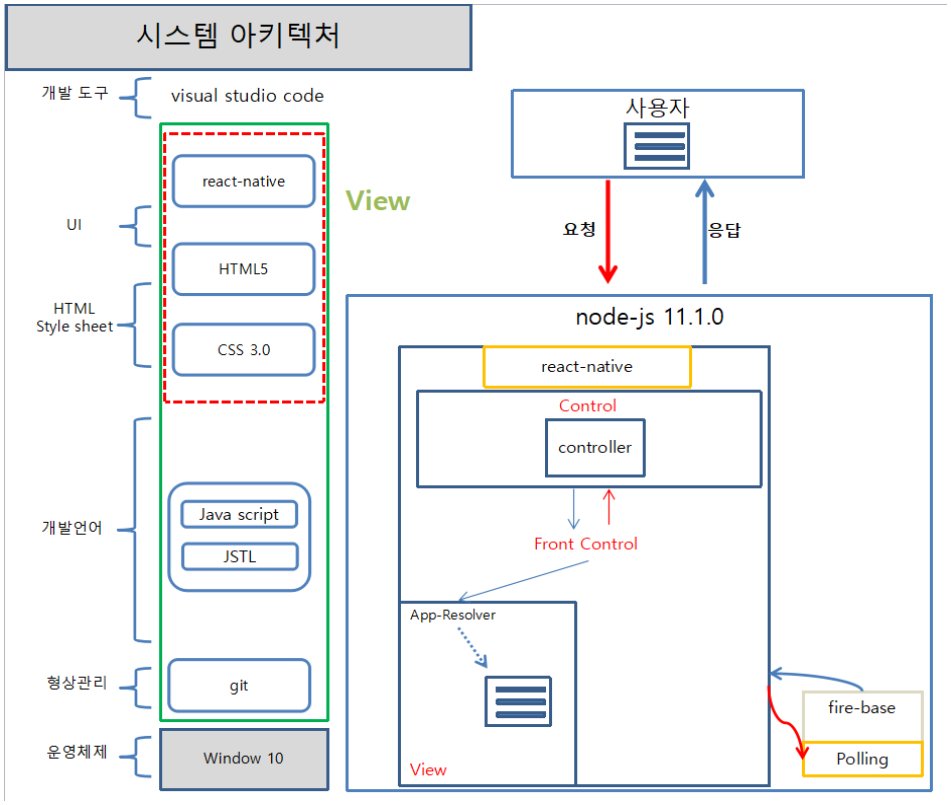
3. **시스템 다이어그램**

1)서버와 데이터베이스 관계도



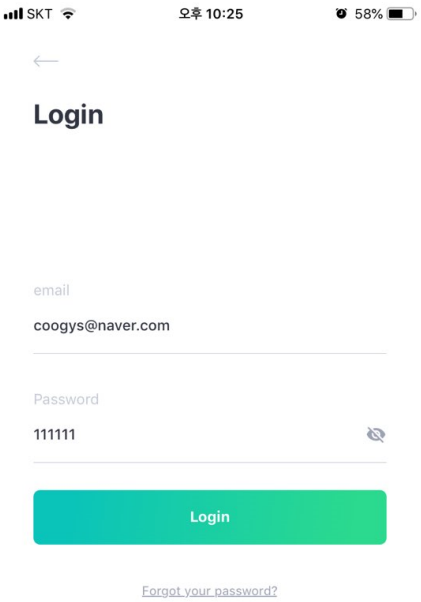
간단하게 요약한 프로젝트 구성도입니다. 서버와 데이터 베이스가 서로 연동되어 데이터를 어플에 전송합니다.

2)시스템 다이어그램


시스템 아키텍처
개발 도구
visual studio code
UI
react-native
HTML5
HTML
Style sheet
CSS 3.0
개발언어
Java script
JSTL
형상관리
git
운영체제
Window 10
View
사용자
요청
응답
node-js 11.1.0
react-native
Control
controller
Front Control
App-Resolver
View
fire-base
Polling

4. 주요 기능

<로그인 기능>



Firebase에서 제공하는 auth기능을 통해 가입된 아이디 확인 , 비밀번호 확인, 그리고 잃어버린 암호 처리하여 이메일로 비밀번호 변경 메일 보내기를 처리하였습니다

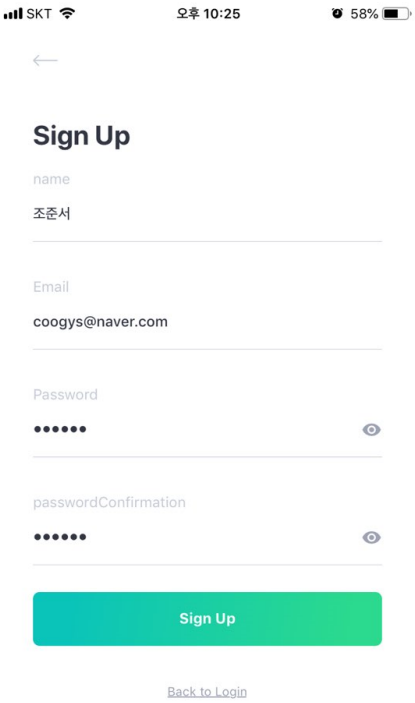
1.로그인 처리 기능

```
firebase.auth().signInWithEmailAndPassword(email, password).then(() => {
        this.setState({ isLoading: false });
        signInApp().then(() => this.props.navigation.navigate('Browse'));
    })
    .catch(() => {
        Alert.alert(
            '아이디와 비밀번호가 옳지 않습니다'
        )
        this.setState({
            isLoading: false,
        });
    });
```

2.비밀번호 찾기

```
api.sendPasswordResetEmail(email)
  .then(() => {
    this.setState({ isLoading: false });
    Alert.alert('email을 확인해주세요.');
    this.props.navigation.navigate('Login')
  })
  .catch((error) => {
    showMessage({
      message: 'Check your form',
      description: `${error.message} (${error.code})`,
      type: 'danger',
    });
    this.setState({
      isLoading: false,
    });
  });
```

<회원 가입>



Firebase에서 제공하는 auth기능을 통해 이메일 중복여부, 비밀번호 최소 개수 지정, 이메일 여부 확인을 구현하였습니다.

1.회원가입 처리 함수

```
firebase
  .auth()
  .createUserWithEmailAndPassword(email, password)
  .then(({ user }) => {
    // Add the new user to the users table
    firebase.database().ref()
      .child('users')
      .push({
        email: this.state.email,
        uid: user.uid,
        name: this.state.name,
      });
    this.setState({ isLoading: false });
    console.log("머야엄야ㅓㅁ");
    return this.props.navigation.navigate('Login');
  }).catch((error) => {
    console.log("error :" + error.code);
    console.log("error :" + error);
    Alert.alert(
      error.message
  )
    this.setState({
      isLoading: false,
    });
  });
```

2. key 값으로 비밀번호 암호화

```
  render() {
    const { navigation } = this.props;
    const { loading, errors } = this.state;
    const hasErrors = key => errors.includes(key) ? styles.hasErrors : null;
```

<게시판>

서버에서 게시글객체배열을 object state로 전부 받아온 뒤, map문에서 제목을 출력하는 컴포넌트인 boardList를 사용해 게시글 목록을 출력합니다.

```
firebase.database().ref().on('value', snapshot => {
  this.setState({ wholeData: snapshot.val()
  });

this.setObject((Object.values(this.state.wholeData))[0])

this.setNumber(Object.keys(this.state.object).length)//게시물 개수 설정

var data=Object.keys(this.state.object);//게시물 키 배열 중 마지막값=(+1해서 글쓰기하면됨)
this.setLastKey(data[data.length-1])

})
```

```
{Object.values(object).reverse().map((info,i)=>{
    return(
      <BoardList title={info.title} content={info.content} writer={info.writer}
      onPressTitle={()=>this.touchTitle(info.title,info.content,
        info.writer,lastKey-i,info.password,info.comment)}/>
    )
})
}
```

게시글 목록 중 제목 하나를 누르면 해당 글의 제목과 내용을 볼 수 있습니다.

React-navigation를 화면전환에 사용했습니다.

```
touchTitle=(text,text2,text3,key,password,comment)=>{
  const {navigate}=this.props.navigation;

  navigate('BoardView',{title:text,content:text2,writer:text3,
    key:key,password:password,comment:comment})


}
```

댓글버튼을 눌러 게시글에 댓글을 작성할 수 있습니다. 첫번째 댓글을 서버에 올릴 때는 그 댓글만 올리면 되지만, 두번째 이상의 댓글은 원래의 댓글+새 댓글 배열로 저장해야 하기 때문에 댓글수에 따라 사용함수를 달리했습니다.

```
this.setState({
    comment:[...this.state.comment,inputComment]
})

firebase.database().ref('board/'+(this.state.key)).update({
  comment : this.state.comment
});
```

```
firebase.database().ref('board/'+(this.state.key)).update({
    comment : [this.state.inputComment]
  });
this.setState({
   comment:[this.state.inputComment]
})
```

수정,삭제 버튼을 누르면 접근자가 입력한 것과 게시물의 작성자가 정한 비밀번호와 일치하는 지 검사한 뒤, 일치한다면 게시글을 수정 및 삭제 할 수 있도록 구현했습니다

```
isCorrect=()=>{
    const {password,inputPassword,key}=this.state;
    const {navigate}=this.props.navigation;
    if(password==inputPassword){
        firebase.database().ref('board/'+(this.state.key)).remove();
        alert('삭제됨')
        navigate('BoardTab')
    }
    else{
        alert('비밀번호가 일치하지 않습니다')

    }
}
```

```
isCorrect=()=>{
    const {password,inputPassword,title,content,key}=this.state;
    const {navigate}=this.props.navigation;
    if(password==inputPassword){
        navigate('BoardModify',{title:title,content:content,key:key})
    }
    else{
        alert('비밀번호가 일치하지 않습니다')

    }
}
```

<정보게시판>

정보게시판은 리액트네이티브의 웹뷰 컴포넌트를 통해 구현했습니다.

<To Do 리스트>

투두리스트의 초기화면입니다. New to do 부분을 눌러 새로운 스케줄을 작성할 수 있습니다.

리액트네이티브에서 제공하는 AsyncStorage API를 이용해 어플리케이션 내부에 목록을 저장했습니다.

```
updateTodo = (id,text) => {
    this.setState(prevState => {
        const newState = {
            ...prevState,
            todos:{
                ...prevState.todos,
                [id]:{...prevState.todos[id],text:text}
            }
        };
        this.saveTodos(newState.todos);
        return {...newState};
    });
};
saveTodos=newTodos => {

   const saveTodos=AsyncStorage.setItem("todos",JSON.stringify(newTodos));
};
```

할일을 완수하고 왼쪽의 동그라미를 누르면 텍스트 스타일이 변경되도록 하여 유저로 하여금 성취감을 느낄 수 있도록 구현했습니다.

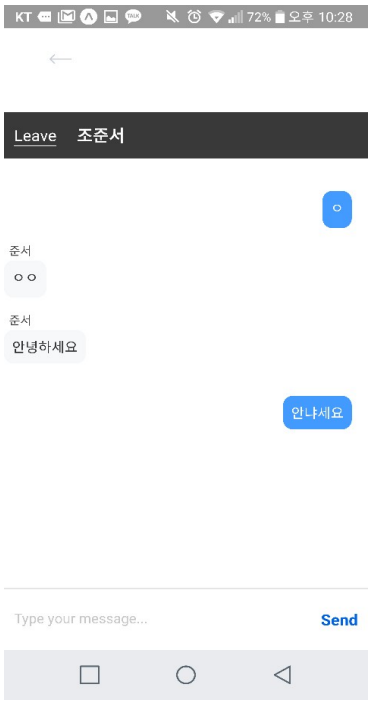
```
completeTodo = id =>{
    this.setState(prevState =>{
        const newState = {
            ...prevState,
            todos:{
                ...prevState.todos,
                [id]:{
                    ...prevState.todos[id],
                    isCompleted:true
                }
            }
        };
        this.saveTodos(newState.todos);
        return {...newState};
    });
};
```

우측의 연필버튼과 x버튼을 눌러 스케줄을 수정 및 삭제 할 수 있도록 했습니다.

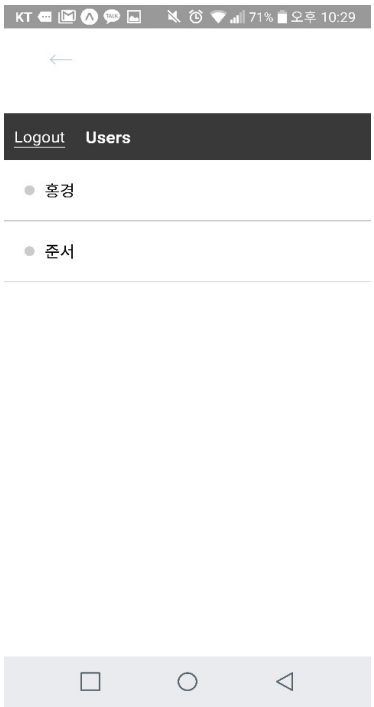
```
deleteTodo = (id) =>{
    this.setState(prevState =>{
        const todos=prevState.todos;
        delete todos[id];
        const newState = {
            ...prevState,
            ...todos
        }
        this.saveTodos(newState.todos);
        return {...newState};
    })
};
```

```
startEditing = (event) => {
 event.stopPropagation();
    this.setState({
        isEditing:true
    });
};
finishEditing = (event) => {
 event.stopPropagation();
    const {todoValue}=this.state;
    const{id,updateTodo}=this.props;
    updateTodo(id,todoValue);
    this.setState({
        isEditing:false
    });
};
```

<채팅 기능>



1:1 채팅 기능을 구현하였고 chatkit 라이브러리를 사용하였습니다.



사용자가 접속하면 오른쪽에 파란 불이 들어오면서 채팅이 가능한 상태가 됩니다.

또한 과별로 톡방이 만들어져 사용자 정보고 얽히는 것을 방지했습니다.

1.사용자 연동 정보, 방 정보 가져오기

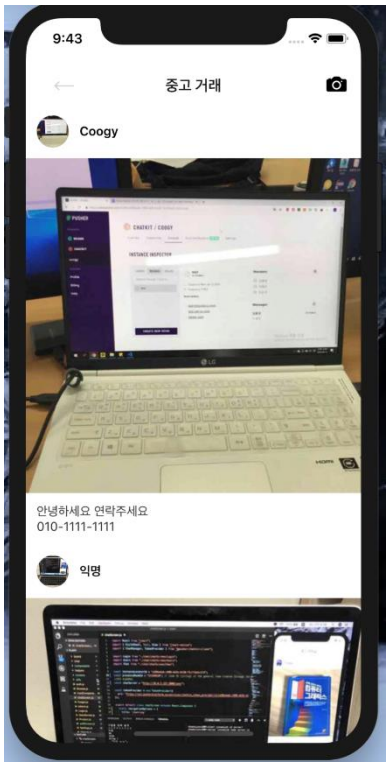
```
const instanceLocatorId = "a206a4ab-c990-4d3e-b230-71c7fda5c174";
const presenceRoomId = "21252441"; // room ID (string) of the general room created through the Chatkit inspector
const chatServer = "http://10.0.3.157:3000/user";
const tokenProvider = new TokenProvider({
  url: "https://us1.pusherplatform.io/services/chatkit_token_provider/v1/a206a4ab-c990-4d3e-b230-71c7fda5c174/token"
});

export default class chatScreen extends React.Component {
  static navigationOptions = {
    title: 'chatting'
  }
```

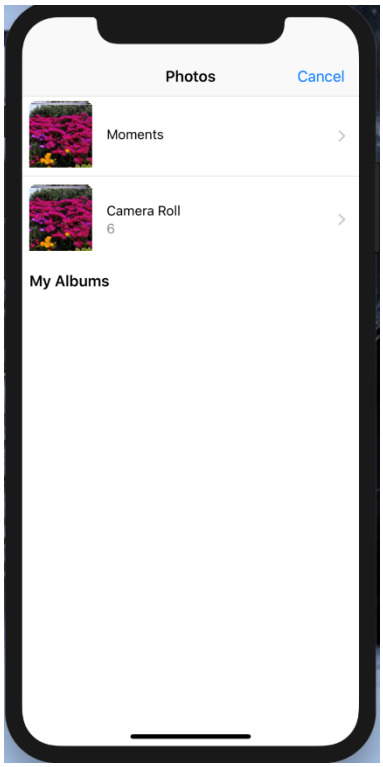
2.메시지 정보 처리 함수, 사용자 정보 확인

```
  renderItem = ({ item }) => {
    let box_style = item.isCurrentUser ? 'current_user_msg' : 'other_user_msg';
    let username_style = item.isCurrentUser
      ? 'current_user_username'
      : 'other_user_username';
```

```
  return (
    <View key={item.key} style={styles.msg}>
      <View style={styles.msg_wrapper}>
        <View style={styles.username}>
          <Text style={[styles.username_text, styles[username_style]]}>
            {item.username}
          </Text>
        </View>
        <View style={[styles.msg_body, styles[box_style]]}>
          <Text style={styles[`${box_style}_text`]}>{item.msg}</Text>
        </View>
      </View>
    </View>
  );
};
```

<중고 거래>





인스타그램 UI 사용하여 사용자 화면에 이미지가 바로 보이게 구현하였습니다.

Firebase사진 저장소와 expo사진 찍기, 사진 저장소 접근 라이브러리를 적용했습니다.

1.이미지 저장과 이미지 확인 함수

```
componentDidMount() {
  if (Fire.shared.uid) {
    this.makeRemoteRequest();
  } else {
    firebase.auth().onAuthStateChanged(user => {
     if (user) {
       this.makeRemoteRequest();
     }
    });
  }
}
```

```
makeRemoteRequest = async lastKey => {
  if (this.state.loading) {
    return;
  }
  this.setState({ loading: true });

  const { data, cursor } = await Fire.shared.getPaged({
    size: PAGE_SIZE,
    start: lastKey,
  });

  this.lastKnownKey = cursor;
  let posts = {};
  for (let child of data) {
    posts[child.key] = child;
  }
  this.addPosts(posts);

  this.setState({ loading: false });
};

_onRefresh = () => this.makeRemoteRequest();

onPressFooter = () => this.makeRemoteRequest(this.lastKnownKey);
```

2.이미지와 글 작성 함수

```
headerRight: (
  <HeaderButtons IconComponent={Ionicons} iconSize={23} color="black">
    <HeaderButtons.Item
      title="Share"
      onPress={() => {
        const text = navigation.getParam('text');
        const image = navigation.getParam('image');
        if (text && image) {
          navigation.goBack();
          Fire.shared.post({ text: text.trim(), image });
        } else {
          alert('Need valid description');
        }
      }}
    />
  </HeaderButtons>
),
```

5. 개발 일정 및 체크리스트

1)개발 일정

	기능	구현내용	7주	8주	9주	10주	11주	12주	13주	14주	15주	16주
회원관리	UI											
	회원가입	이메일, 이름, 비밀번호 입력받음 -> 받은 정보를 DB 유저풀에 저장										
	로그인	이메일, 비밀번호 입력받음 -> DB에 저장된 정보 검색 및 검사										
	로그아웃											
	비번찾기	이메일 입력 후 DB에 저장된 정보 검색 및 검사 -> 확인 메일 전송										
회대소식		웹뷰구현										
자유게시판	UI											
	게시물조회	DB에서 모든 게시물 객체 fetch -> 게시물 제목 출력										
	게시물등록	제목, 내용, 비밀번호 입력받고 DB저장										
	게시물수정	비밀번호 검사 -> 수정된 제목과 내용을 DB에 업데이트										
	게시물삭제	비밀번호 검사 -> 해당 게시물 DB에서 삭제										
	댓글조회	DB에서 해당 게시물의 댓글배열 fetch -> 댓글 출력										
	댓글등록	내용 입력받고 기존 댓글 배열의 끝에 저장 -> DB업데이트										

2)체크리스트

	기능	완성도	대체제
회원관리	UI	○	
	회원가입	○	
	로그인	○	
	로그아웃	X	
	비번찾기	○	
정보게시판	api 가져오기	△	웹뷰 구현
자유게시판	UI	○	
	게시물조회	○	
	게시물등록	○	
	게시물수정	○	
	게시물삭제	○	
	댓글조회	○	
	댓글등록	○	
	댓글수정	X	
	댓글삭제	X	
	검색	X	
	작성일 조회	X	
	작성자 조회	△	익명처리
채팅	UI	○	
	서버	○	
	단체채팅	X	
투두리스트	UI	○	
	투두등록	○	
	투두수정	○	
	투두삭제	○	
	투두완료	○	

	데이터베이스	○	
장터	UI	○	
	카메라	○	
	앨범	○	
	사진등록	○	
	게시글등록	○	
	글수정	X	
	글삭제	X	

6. 수정 및 보완점

체크리스트에서 체크가 안 된 부분들을 우선적으로 수정하고 보완할 것이고, 각 기능 탭의 UI가 단조로우므로 더 보기좋게 다듬을 것입니다. 더 나아가 기능들을 계획대로 구현하지 못했거나 대체방법을 쓴 것으로는 투두리스트를 파이어베이스에 연동하지 않고 AsyncStorage를 사용하는 것에 그친 것, 게시판의 댓글 수정 및 삭제기능을 구현 못한 것, to do에 날짜별 관리 기능을 더하지 못한 것이 있는데 이 부분의 작업을 통해 어플리케이션을 발전시켜볼 계획입니다.

회원관리 측면에서는 로그아웃 기능이 없기 때문에 로그아웃 기능을 만들것입니다.

정보게시판 같은 경우에는 처음에 페이스북 API를 사용해 간단히 화면에 뿌려주는 것을 기대했지만, API 공부가 미흡하여 리액트 네이티브의 기능인 webview를 통해 페이지를 불러왔습니다. 이 과정을 통해 API를 적용하는 것에도 내공이 필요하다는 것을 깨달았고, 더 공부하여 페이스북API를 사용해 기능을 구현하도록

노력할 것입니다.

자유게시판은 기능 구현에만 열심을 낸 까닭에 UI가 미관상 부족한 점이 있어 보기좋게 다듬을 필요가 있습니다. 그리고 로그인 연동 측면에서 자유게시판 탭과 파이어베이스를 상호작용 시키는 것에 어려움을 느꼈기 때문에 비밀번호를 입력하는 기능을 추가했고, 글 작성자가 익명이 아닌 로그인한 회원이름으로 저장 될 수 있도록 노력할 것입니다. 댓글은 작성기능만 있으므로 수정, 삭제 기능을 추가해야합니다.

채팅 탭의 경우는 서버의 불안정성을 해결하지 못한 문제가 있었기에 아쉬웠고, 단체채팅 구현을 한다면 더 완성도 있는 어플리케이션이 될 것 같습니다.