Team Information

- 14' Kim Seongmin
- 13' Lee Jaehong
- 15' Choi Jihyeok
- 15' Kim Sebin
- 15' Song Joonho

Product Design Specification

Name of the product

- Software Development Project Management Tool

Architecture

1. Major modules and functionalities

1) Schedule Management(enrollment / configure / deletion)

- User(Team member) can enroll their schedule to calendar. the schedule information will be synchronized to task list. other team member can view the tasks.

2) Calendar

- User can view and enroll their schedule. the information will be also enrolled to task list. calendar will be give user visibility for many tasks.

3) Task List

- we have dashboard to notify "to do", "doing", "done" tasks for everyday for user.

4) Search

- One user can search the schedule of other user. so, team members can communicate with shared calendar and tasks list.

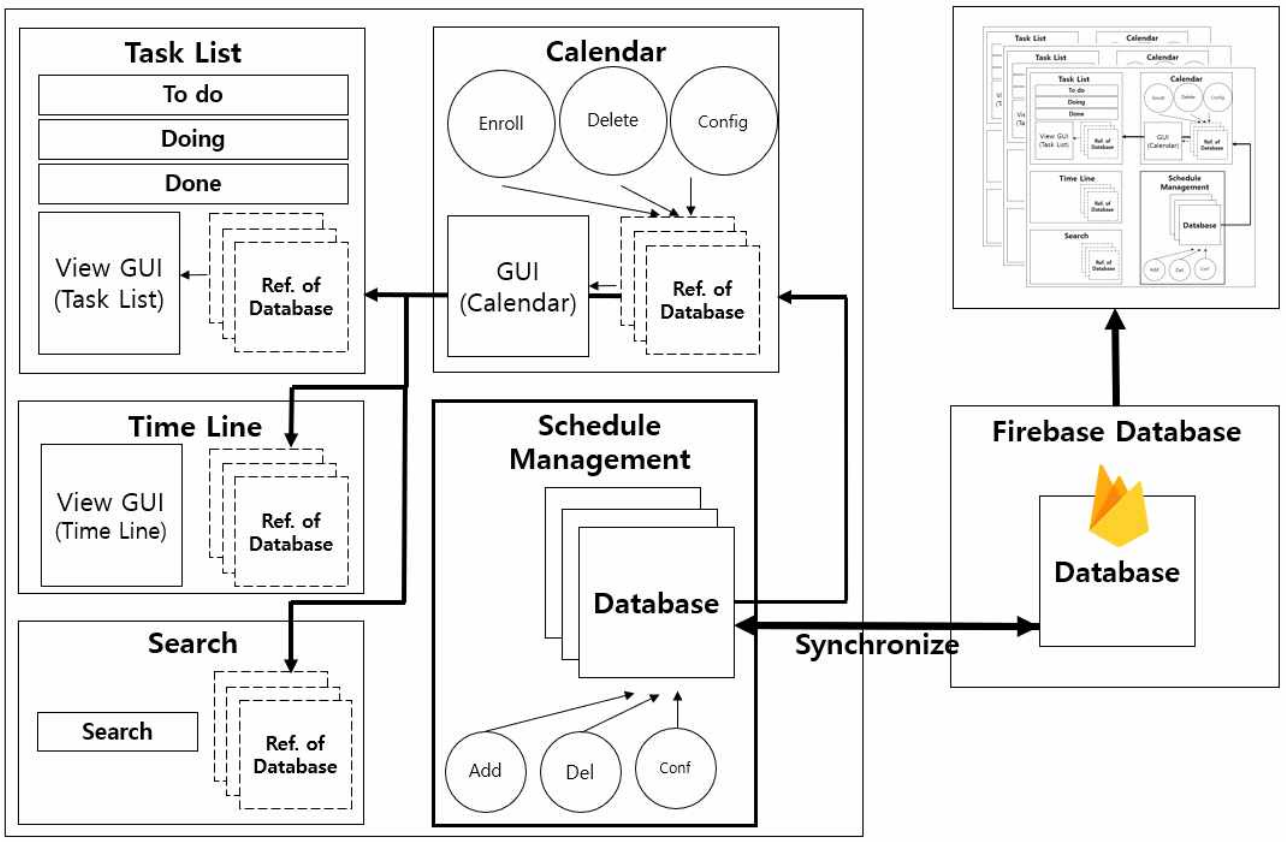
5) User can register milestones to their project. (Task List)

- Milestone means sub-project contained to main project. so, they can view their objectives and also team member's objectives. and they can register their schedule to their milestone.

6) Timeline diagram

- Timeline can provide visibility for topologically sorted tasks for every team member.

2. Interfaces between modules



1) **Schedule Management** is local database of schedules. So It has right to access schedule data by adding, deleting, configuring them. It Just send database reference to **calendar.** and Schedule Management try to update and synchronize data from Remote Firebase database.

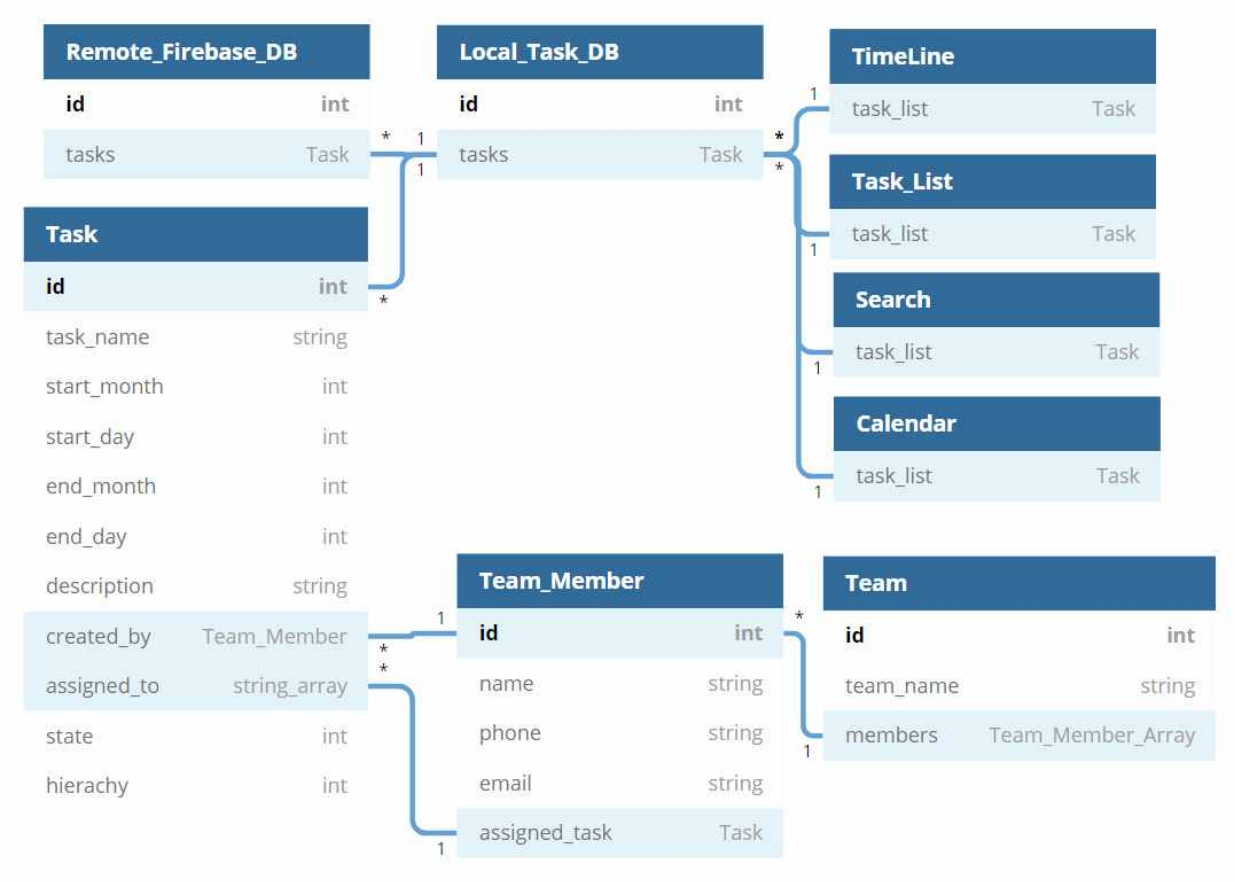
2) **Calendar** receive reference of schedule database from **Schedule Management**. and Calendar enroll data to its Graphic User Interface.

3) **Time Line** and **TaskList** receive reference of schedule database from **Schedule Management** and update TaskListView.

4) **Search** module receive reference of schedule database. and It has search method to find some schedule what user want to find.

5) **TaskList, Time Line, Search, Calendar** can't modify and add, delete data by all means. If the function is needed, they have to call **Schedule Manager**.

3. Data description (database schema)



4. Design alternatives

Most of software development project has many sub-project needs and has to be managed by project manager. However, no matter how talented project manager is in the team, he/she can't manage their whole team projects efficiently without scheduler or proper management tools. Our project management tool provides you the ability to establish a hierarchy of tasks easily for completion of effective project completion. and it will gives order to tasks which are dependent on each other.

Anyone can easily know that the project can't be completed well without cooperation among coworkers. If communication among co-workers isn't good enough to know the schedule or progress of on going project, the relationship between colleagues will get worse and the project may fail. Therefore, project management is needed to solve this problem.

In this way, Software development progress management tool can help manager to complete large project by making to share critical issues, communicate among peers, track progress of project, make plan easily.

Process

1. Risk Assessment

● **Estimation and Scheduling mistake**

The unique nature of individual software projects creates problems for developers and managers in estimating and scheduling development time

● **Not planned requirements growth**

issues that are not identified earlier can create a last-minute hurdle to meeting deadlines

● **Breakdown of specification**

During the initial phases of integration and coding, requirements might conflict. Also, developers can find that even the specification is unclear or incomplete.

● **Technical risks**

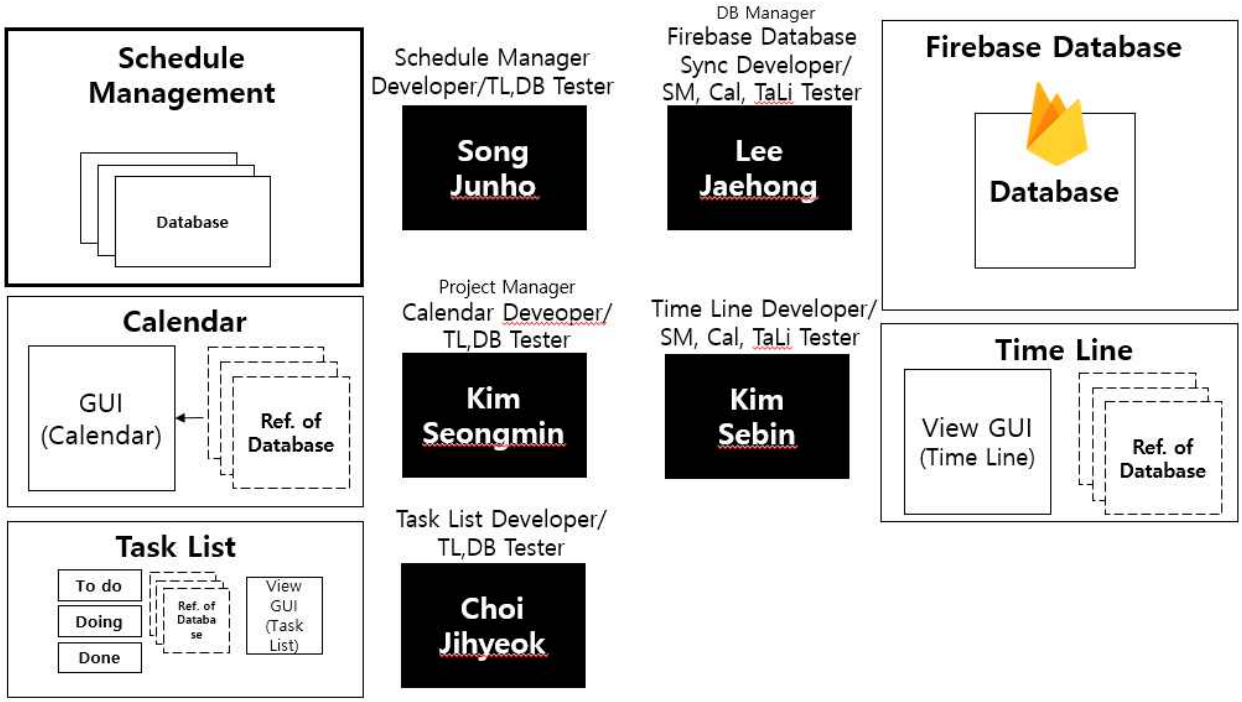
Software development projects can be reduced the functionality of the software to compensate for overruns pertaining to high budgets and scheduling. There can be conflicts between achieving functionality of the software.

2. Project schedule

Week / Contents	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Proposal	■	■	■					Midterm Exam								Final Exam
Requirements				■	■											
UML Design (virtual)						■										
Schedule Management (Add,Config,Del)						■	■		■							
Calendar Function (Add,Config,Del)						■	■		■							
Task list Function (Add,Config,Del)						■	■		■							
Testing (Schedule, Calendar, Task list)										■						
Synchronization between functions											■	■				
Extension to multiple team member											■	■				
Timeline Diagram										■	■	■				
Testing (Syncronization, Extension, Time Line)													■			
Design (Color, Button, etc.)													■	■		
Report Documentation														■	■	

3. Team structure

	Kim Seongmin	Lee Jaehong	Choi Jihyeok	Kim Sebin	Song junho
Developer	O	O	O	O	O
Tester	O	O	O	O	O
Designer		O	O		



4. Test plan

Testing plan in included in schedule list. We use agile method to develop this project management program. so, we will do **component test** after develop small components. and after develop all functions, We will do **function connection tests** with component tests what we've done before. and We will use **JUnit Library to** test our program. (that`s why we chose Java for main language)

Finally, We will do Final Test with component test and function connection tests after development is completely done.

5. Documentation plan

As we said, We are using agile method to develop this program. We need to minimize documentation. so, we are going to write short comments on the top of source code. After function connection level, we will use "How to

use" and "Development Document for simple explanation of data and method and how it works.

6. Coding Style Guidelines

Structure

× Classes should be named with uppercase letters.
× Classes should only do one specific task.
× Classes should not just hold methods but also data.

Coding

× There should be minimal spacing between words (ex. HandsomeJoonHo)
× Long lines of code should not be split up unless it provides clarity to the code
× Comments should be used as much as possible, they should provide clarity to the code, or notes to future developers
× Comments should start with a space and then a capital letter

× Javadoc double comments should be added to every method to provide clarity when referenced from other classes
× Javadoc comments should have all return and parameter statements included and explained what their states/purpose is

× Conditional statements should be on one line for clarity and always have a comment in a human readable format above it.
× Declarations should be made on multiple lines, even if they can be shorten to one line.

```
// Example Declaration
BigDecimal foo = new BigDecimal(0), bar = new BigDecimal(0);
// Corrected
BigDecimal foo = new BigDecimal(0);
BigDecimal bar = new BigDecimal(0);
```

× Tabs should be 4 spaces. this allows easy editing and viewing on github

A efficiency program will always be of higher priority. Even if something can be done in a day, take the time to make a program/logic that will last more then the time expected, and that will take into account unseen errors
Make your program "flow" it should run from the top to the bottom, a person should be able to read down the page and understand what is happening
The main class is for starting the program object, not for creating everything.
The main should have the most comments, when someone reads through the main they should be able to understand what the whole program does
This repo is for the public, even if it was private a programmer should code to make his code look just as beautiful as it is in functionality
Don't write BS code, take the time to make something worth committing
× Have clean commits, if you are going to do formatting, do it in another commit. Other contributors should be able to see clean diffs and understand what is happening
× Commit messages should be descriptive, while the commit info should have details

Example

```
import java.util;

public class UndercastClient {

  // Data storage
  public static HashMap<String, Object> client_data;

  /**
   *  Default Constructor for the Undercast Client Mod.
   */
  public UndercastClient() {
    // TODO: Load data from local storage if needed
    // Create data hash
    client_data = new HashMap<String, Object>();
    // Defaults
    this.setDataDefaults_All();
  }

  /**
   * This method starts all listeners that the client uses.
   * Each Listener can be configured in the config
   */
  public void startListeners() {
    // Main display listener
    new DisplayListener();
    // Data update listeners
    new DataListener();
    // TODO: Background thread listeners
  }

  /**
   * This is just an example method.
   * These should be descriptive
   *
   * @param message What message was received
```

```
   * @param noColorMessage What messaged was received minus the color codes
   */
  public static void handleMessage(String message, String noColorMessage) {

    // Cancel the event if needed
    if(line_capture != 0){
      line_capture -= 1;
      event.setCanceled(true);
    }

    // Server current update
    if(noColorMessage.contains("You are currently on ")){
      String server = noColorMessage.replace("You are currently on ", "");
      UndercastClient.client_data.put("server", server);
      event.setCanceled(true);
      line_capture += 1;
      return;
    }

    return;
  }
}
```

Team Website

- gitHub Organization
- Website : https://github.com/pm-tool-se2019