**Attendance Tracker**



Software Design Document

Version: 1.0

11.11.2018

MertaliKöprülü

Metin Berk Karataş

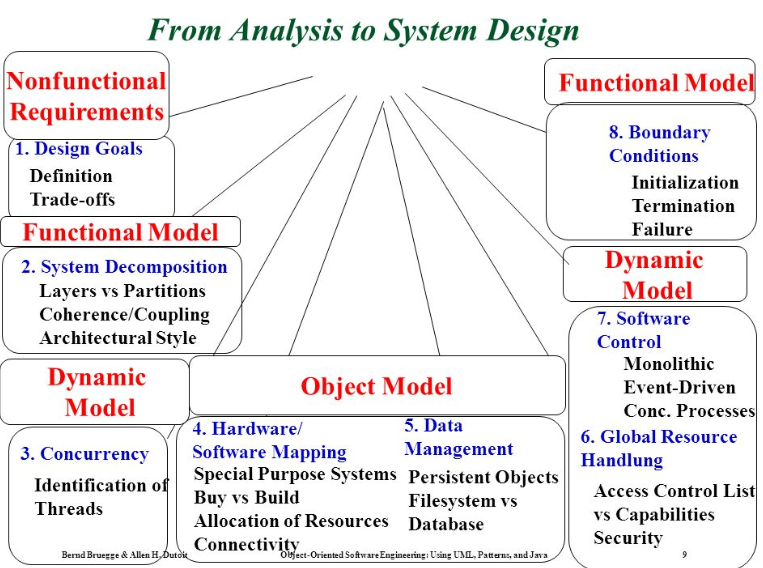
Gülden Güllü

Korhan Köz

Prepared for

SE301 Software Engineering





|  |  |  |  |
| --- | --- | --- | --- |
| Topic: | Name: | Topic: | Name: |
| 1.1.Purpose of the System | + | 3.3. Hardware Software Mapping | (Korhan) |
| 1.2.Design Goal | (Korhan) | 3.4. Persistent Data Management | (DBER)(Mertali) |
| 1.3. Definitions,  Acronyms, and Abbreviations | + | 3.5 Access Control and Security | Gülden + |
| 1.4. References | (Korhan) | 3.6. Global Software Control | Gülden + |
| 2.Current Software Architecture | (Berk) | 3.7. Boundary Services | Gülden + |
| 3.1.  Overview | (Korhan) | 4. Subsystem Services |  |
| 3.2.  System Decomposition |  | 5.References | + |

Table of Contents

[1. Introduction 1](#_Toc433996772)

[1.1. Purpose of the System 1](#_Toc433996773)

[1.2. Design Goals 1](#_Toc433996774)

[1.3. Definitions, Acronyms, and Abbreviations 1](#_Toc433996775)

[1.4. References 1](#_Toc433996776)

[2. Current Software Architecture 1](#_Toc433996777)

[3. Proposed Software Architecture 1](#_Toc433996778)

[3.1. Overview 1](#_Toc433996779)

[3.2. System Decomposition 1](#_Toc433996780)

[3.3. Hardware Software Mapping 2](#_Toc433996781)

[3.4. Persistent Data Management 2](#_Toc433996782)

[3.5. Access Control and Security 2](#_Toc433996783)

[3.6. Global Software Control 2](#_Toc433996784)

[3.7. Boundary Conditions 2](#_Toc433996785)

[4. Subsystem Services 2](#_Toc433996786)

[5. References 2](#_Toc433996787)

SYSTEM DESIGN DOCUMENT[Error: Reference source not found]

The System Design Document (SDD) is written after the initial system decomposition is done, and updated throughout the development. SDD describes the services provided by each subsystem. Although this section is usually empty or incomplete in the first versions of the SDD, this section serves as a reference for teams for the boundaries between their subsystems. The interface of each subsystem is derived from this section and detailed in the Object Design Document.

SDD is used to define interfaces between teams of developers and serve as a reference when architecture-level decisions need to be revisited. The audience for the SDD includes the project management, the system architects (i.e., the developers who participate in the system design), and the developers who design and implement each subsystem.

1. Introduction

Provide a brief overview of the software architecture and the design goals. It also provides references to other documents and traceability information (e.g., related requirements analysis document, references to existing systems, constraints impacting the software architecture).

* 1. Purpose of the System

Aim of the system is to make attendance track simple, quick and trackable by the student as well. It will be simple learn for instructor because it has many options that are used in today’s pen-paper method with extensions of being in digital like getting whole data at a click of finger in the of semesters. Also, it will be quicker than writing every name in the class or checking every student on class. It will be a mobile application, so it will be easier than table using system. In addition, students will be able to track their attendance so that they will be able to know their status about attendance.

* 1. Design Goals
* **Response Time:**

User's requests or actions must be acknowledged fast and put in to action as quick as possible, because one of "AWSHEET's main goal is to save time from user's daily university life.

* **Robustness:**

Because of the usage field of our app, AWSHEET needs some serious amount of input from users for certain functions. Users can give invalid input and AWSHEET must avoid this by giving enough information to user and also having such input fields so that the input errors that can be caused by the users are minimized.

* **Reliability:**

System has to continue operations that user requests without errors. Reliability of the system should be high and the data taken from user should be retained securely.

* **Security:**

AWSHEET must retain the important data that belongs to user successfully. Otherwise this may cause real life problems that affect Instructors and Courses in a bad way. This must be avoided so authentication system should work perfectly and the password or personal information of users must be encoded.

* **Portability:**

In future stages of AWSHEET it can be easily applied to IOS platform too, this will increase the number of our potential clients.

* **Extensibility:**

In future stages of AWSHEET new functions can be added with requests and feedbacks from users. While our first priority isn't social activities, system can support any function that involve Instructors Assistants and Students in a university.

* **Usability:**

Functions that are supported in AWSHEET are mostly actions that needed to be done by our users in real life. So they won't feel strange using the app and the learning process will be real quick. Also since one of our main goals is to make these everyday actions simpler and quicker our users won't feel stressed and confused while using AWSHEET.

* 1. Definitions, Acronyms, and Abbreviations

AW Sheet: Attendance Watch Sheet

* 1. References

There isn't an exact existing system for us to replace. But used system that resembles what AWSHEET provides for user is Course Online. Course Online's purpose and functions is a lot bigger in range. It provides all the university communication among Instructors, Assistants and Students. Courses are taken from that system; homework and project assignments are given and submitted from that system etc. Because it covers a large area of functions usability of it decreases thus making it not easy to use. Or we can say in other words, not a smooth experience. AWSHEET contains some of the same functions as Course Online but one of AWSHEET's purposes is to make things easier and quicker for the users.

1. Current Software Architecture

The current similar system has two main architectural styles which are modal/view/controller and client/server. MVC is the first one and its is used widely on websites like current system. It can make dynamic views for the same modal by using controller or by directly accessing it. Second style the current system uses is client/server which also widely used style where users uses client to request any services. The current uses HTTPS type of this style.

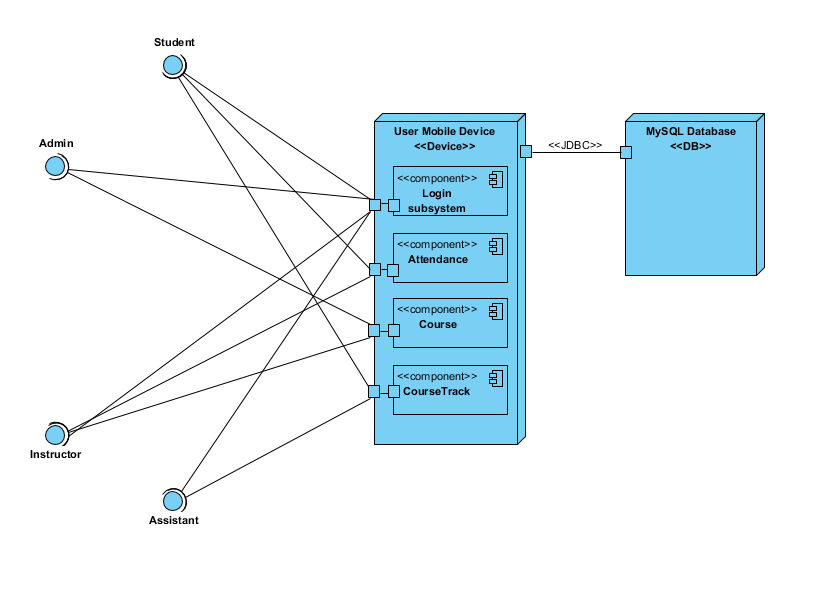
1. Proposed Software Architecture

Documents the system design model of the new system.

* 1. Overview
* **Student Interface:** This Interface provides services that all students need to use in this application. This Interface is planned and implemented according to our Design Goals for giving a smooth and easy experience for the user. Also provides the forms needed to be filled by students when the specific functions are running. For example; Calculate Grade, Attend Course, Calculate Attendance.
* **Instructor/Assistant Interface:** This Interface provides services that Instructors and Assistants need to use in this application. This Interface is planned and implemented according to our Design Goals for giving a smooth and easy experience for the user. Also provides the forms needed to be filled by Instructors or Assistants when specific functions are running. For example; Request Grade, Enter Grade, Take Attendance, Give Attendance.
* **Admin Interface:** This Interface provides services that Admin need in order to administer and manage the system. All the forms needed to be filled by admin are included in this Interface. For example; Add Instructors, Add Assistants, Add Course Lists.
* **Login Subsystem:** This subsystem contains Login,Register,Password Reset and User Authentication services. Some of this subsystem's functions can be customized according to user's needs
  1. System Decomposition

Describe the decomposition into **subsystems and the responsibilities** of each. **This is the main product of system design.**

* 1. Hardware Software Mapping



* 1. Persistent Data Management

Describe the persistent data stored by the system and the data management infrastructure required for it. This section typically includes the description of **data schemes, the selection of a database, and the description of the encapsulation of the database**.

* 1. Access Control and Security

Access Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **.** | **System** | **Attendance** | **Course** | **Announcement** | **Grading** |
| **Instructor** | login() logout() forgetPassword() resetPassword() | takeAttendance() requestAttendanceList() | editCourse() selectCourse() selectAssistant() | sendAnnouncement() checkAnnouncement() | enterGrade() requestGrade() |
| **Assistant** | login() logout() forgetPassword() resetPassword() | takeAttendance() sendAttendanceList() |  | sendAnnouncement() checkAnnouncement() | enterGrade() sendGrade() |
| **Student** | register() login() logout() forgetPassword() resetPassword() | participateInAttendance() trackAttendance() | attendCourse() withdrawCourse() | checkAnnouncement() | calculateGrade() trackGrade() |
| **Admin** | login() logout() forgetPassword() resetPassword() addCourses() addUser() deleteUser() determineAddDropPeriod() |  |  | sendAnnouncement() checkAnnouncement() |  |

The users of the AW Sheet App will log in to the system by using their e-mails and passwords. The admin user will be initialized by the developers, e-mail and password will be determined. Instructor and assistant users will be created by admin, auto- generated password will be sent to their e-mails. Students will register to the systems using their e-mails and creating a password. Users always can change their passwords to keep secure. In database, passwords will be kept encrypted. No user can access real string of passwords. The user type check will be done from the database “User” table, user type attribute.

* 1. Global Software Control

The AW Sheet app will be implemented with decentralized control with threads since we have an object-oriented program, multi-step sequences and users connecting to the service at the same time. With increased number of users, requests at an instance will increase and bottleneck should be avoided as much as possible. Because multiple users may access to database, to avoid corruption of the data, synchronization of multiple threads will be made so only one thread can access the resource at a given point in time.

* 1. Boundary Conditions

The AW Sheet system will start-up by admin clicking “Initialize System” button, the system must be disconnected before. With the initialization, users may register or log in to the system. Admin can modify the settings for the system such as adding/deleting users or changing add/drop semester period. After the initialization of the system, except some special functions which require date constraints such as add/drop course or withdraw course, all allowed functions for the users will be available.

Within the implementation, all predicted errors will be ready to be handled. Instead of terminating the application, the error handlers will let the user know what went wrong and how to fix the error with a feedback. As an example, if the user disconnects to the system when trying to perform an action, the system will show an alert box to user letting know that the user has disconnected and the action could not be completed. However, change requests of the database will be saved instantly to keep the data loss or corruption minimum.

To shutdown the system, the admin uses “Shutdown System” function. All logged in users will be logged out and disconnected to the system. Data saved to the database until the last request before the termination will be protected during the termination process. The termination may possibly be done during maintenance processes and out of the semester periods.

|  |  |
| --- | --- |
| *Use Case Name* | StartServer |
| *Participating actor*  *Instances* | Initiated by Admin |
| *Flow of events* | 1. Admin clicks the Initialize System button. 2. If it is the starting of the semester, admin must enter the semester period, add/drop period, course list and at least one instructor user. |
| *Entry condition* | Admin is logged into the system successfully. |
| *Exit condition* | AWSheet system has been initialized and can be used by users. |

|  |  |
| --- | --- |
| *Use Case Name* | ShutdownServer |
| *Participating actor*  *Instances* | Initiated by Admin |
| *Flow of events* | 1. Admin clicks the Shutdown System button. 2. If it is the end of the semester, all courses’ information, add/drop period dates and semester dates should be deleted from the system. |
| *Entry condition* | Admin is logged into the system successfully. System server has started before. |
| *Exit condition* | AWSheet system has been terminated and users are disconnected. |

|  |  |
| --- | --- |
| *Use Case Name* | MissingGradesException |
| *Participating actor*  *Instances* | *Initiated by Student* |
| *Flow of events* | 1. Student has not entered all of the required grades. 2. Student taps to “Calculate Grade” button. 3. System displays an alert box telling “Missing Grades! Final Grade could not be calculated.” 4. Student confirms the alert box. 5. Student is on the Calculate Grade function interface again. |
| *Entry condition* | Student is logged in and using “Calculate Grade” function. |
| *Exit condition* | Student either calculates final grade or exits the function. |

1. Subsystem Services

Describe the **services provided by each subsystem**. Although this section is usually empty or incomplete in the first versions of the SDD, this section serves as a reference for teams for the boundaries between their subsystems. The interface of each subsystem is derived from this section and detailed in the Object Design Document.

1. References
2. “Itslearning App.” *Itslearning - Global*, itslearning.com/us/itslearning-app/.