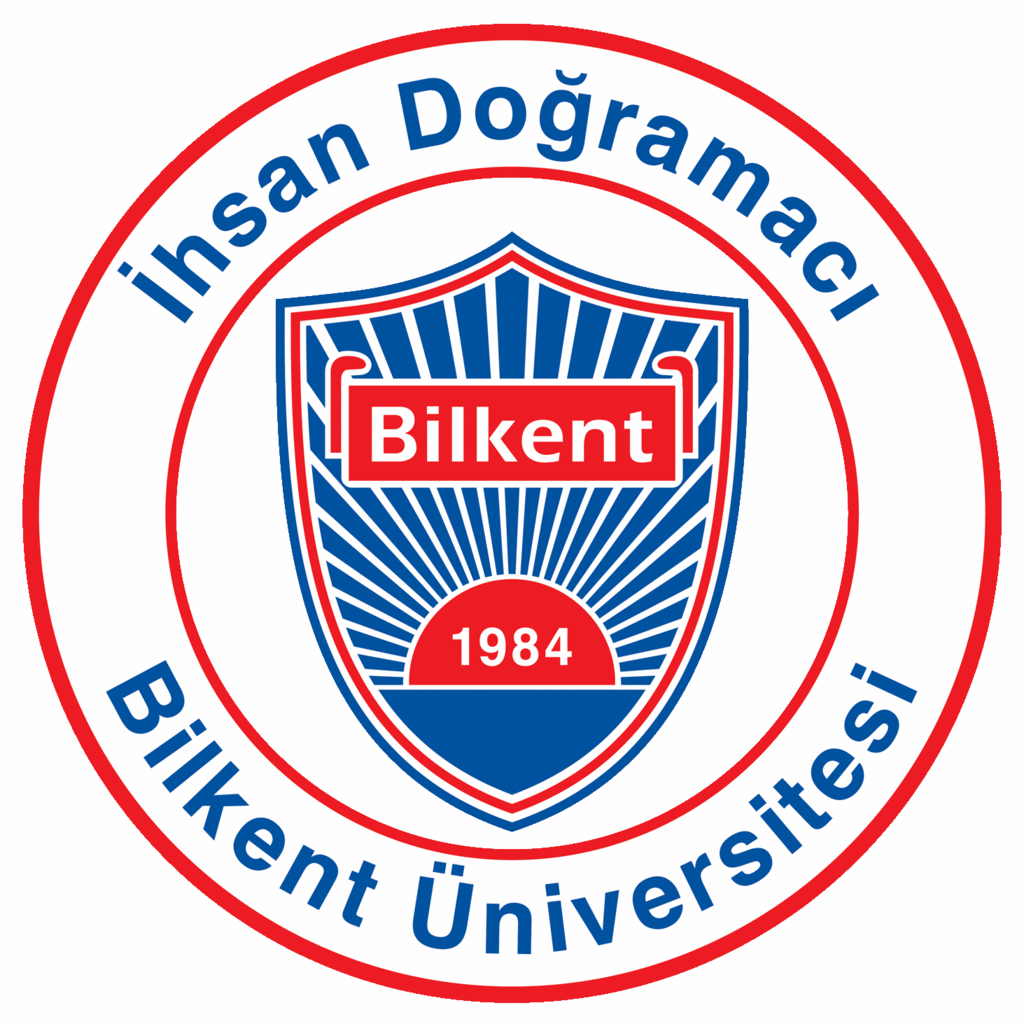
****

Bilkent University

Department of Computer Engineering

**Software Architecture Design**

**Useful ProDers App for Headmaster of Schools**

**Final Report**

**Group-16**

**Ahmet Ay**

**Anıl Göllü**

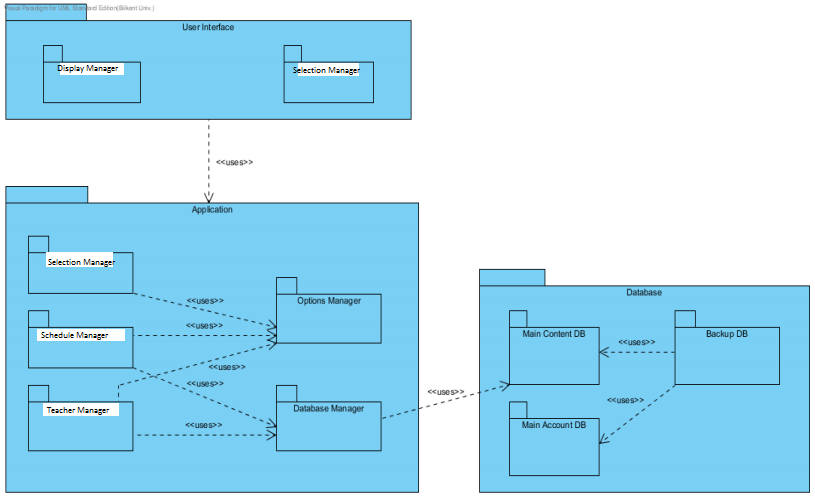
**Cüneyt Erem**

**Muratcan Katırcı**

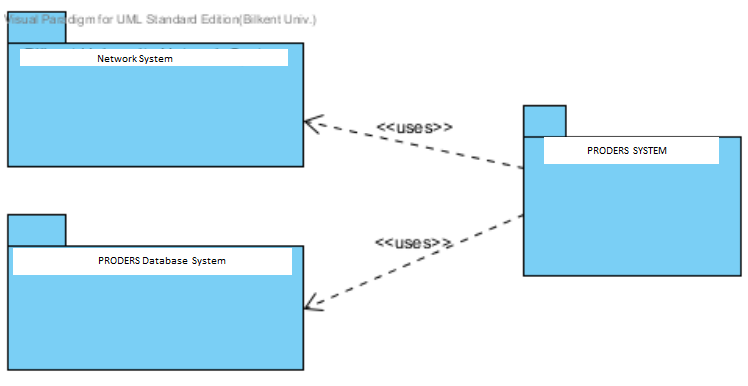
**9. Module Views**

Module views of the “PRODERS” web-based application will be demonstrated in this part. The principal units of implementations of the application are module views. Thus, the elements of these view modules will be used. Modules are the implementation units of software that provides an easy and well understandable unit of functionality.

**9.1 USES VIEW**

****

The figure above represents the uses view of PRODERS. Generally, Application Module is used by User Interface Module and uses Database Module. User Interface Module includes Display Manager and Selection Manager Packages. In Application Module, Selection Manager uses Options Manager package, while Schedule and Teacher Manager Packages use both Options Manager and Database Manager Packages. In order to make any transaction in the system regarding to Schedule or Teacher, these packages have to access the Database. Database Manager that Schedule and Teacher Manager use uses Main Content DB package of Database Module. This module again is used by Backup DB to save or recover any data in any unexpected situation. Backup DB package also uses Main Account DB package.

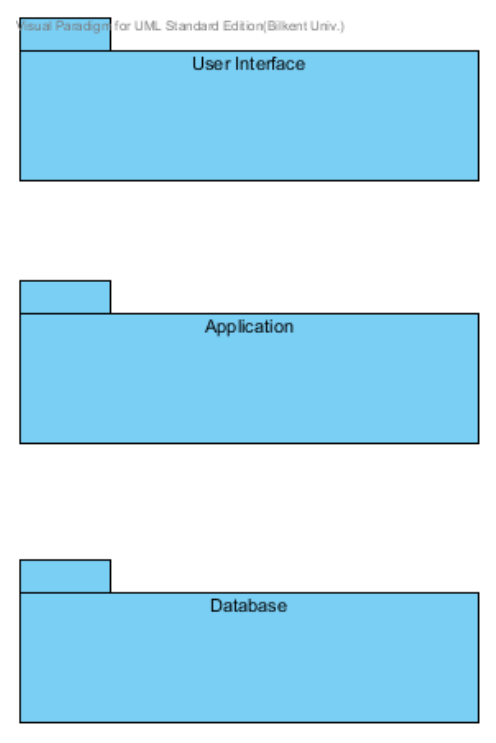


**Context Diagram of Uses View**

Context diagram for uses view demonstrates the external entities that use or being used by the system under it has been developed. Proders application’s Context Diagram is shown above. PRODERS system uses two external systems which are Network System and PRODERS Database System.

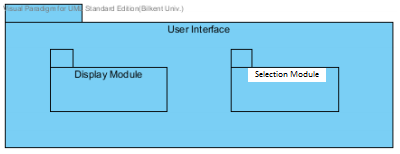
**9.2 DECOMPOSITION VIEW**

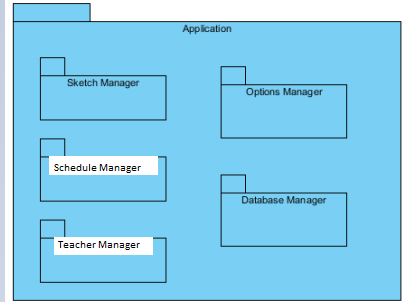
The top level decomposition view of PRODERS is represented in the figure below. There are three main modules in the system. These are User Interface Module, Application Module, and Database Module. If we proceed deeper inside the User Interface Module, we face GUI and PHP related modules that mentions about the visual properties of the system. Database Module is where all the transactions and datas are kept so that it is a kind of source of the system. The most part of the system is done in Application Module.

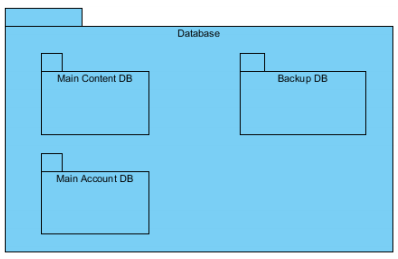


**Top Level Decomposition View of PRODERS**

Decomposition view of the PRODERS consists of three main modules which are called User Interface, Application and Database, and the modules in each of them. User Interface Module consists of Display Module, where buttons, checkboxes, images and multimedia related user content are provided to be visualized and Selection Module, where users log in the system as user or admin and decides to what to change or add suchs as creating a new schedule, class, teacher or editting an existing one of them. Such opportunities are visualized in Selection Module. In Application Module, there are Selection Manager, Schedule Manager and Teacher Manager which control selection applications, transaction applications, creation applications and multimedia applications respectively. Furthermore, there is Options Manager in Application module which controls or changes the options of the system such as font size of the scripts, background colors, line colors of the schedule, language etc. Finally, there is a Database Manager which is responsible for retrieving or sending information to database. In the Database Module, there are two main databases which are Main Content DB and Main Account DB, in addition to these there is Backup DB to recover the two databases in unexpected situations.

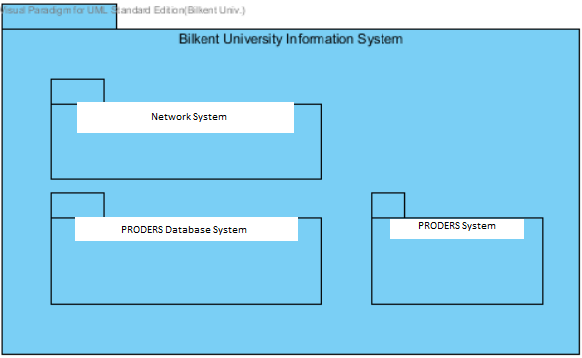
****

****

****

**Decomposition View of PRODERS**

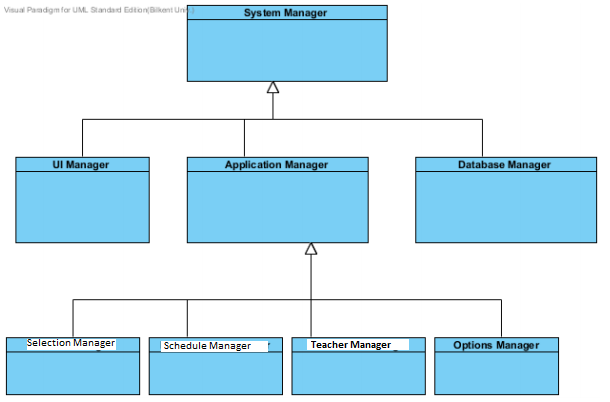
Context diagram for decomposition view represents “is a part of” relation between a larger system and its subsystems. PRODERS is a web-based application which can be represented by context diagram as shown in Figure below. In that system, there are school Network System and school Database System in addition to Application.



Context Diagram of Decomposition View

**9.3. GENERALIZATION VIEW**

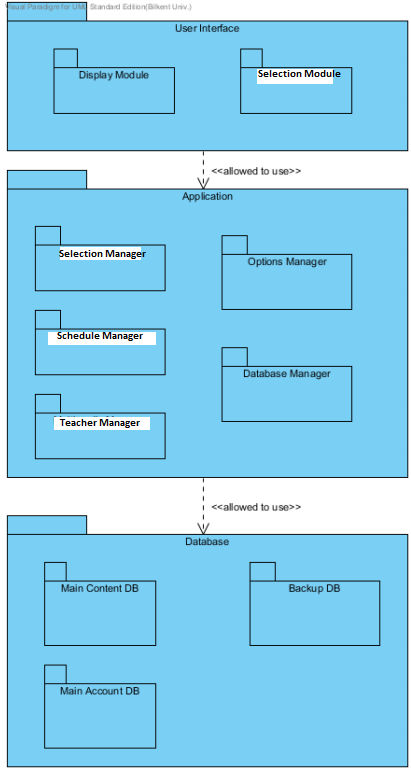
The generalization view of PRODERS is illustrated at Figure below this paragraph. The system has Selection Manager, Schedule Manager, Teacher Manager and Options Manager which are the extension of Application Manager. Moreover, Application Manager is the ancestor of those managers. Also, User Interface Manager, Application Manager and Database Manager are the subsystems which control related parts of the system; these are the extensions of System Manager.



Generalization View of PRODERS

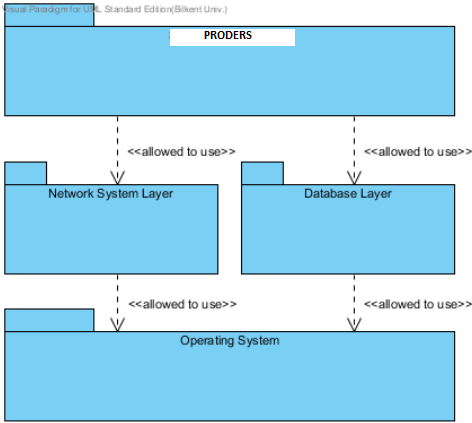
**9.4. LAYERS VIEW**

Layered view is benefitted to separate the modules of PRODERS and define their permission to use relationships for each other. Layered view shows the PRODERS's permission to use different module together and within them.This type of view is important for separation of troubles and modularity of the system. The elements of Layered view are the layers and their relationships are allowed to use relations.

****

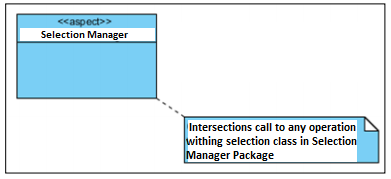
Layered View of PRODERS

PRODERS involves three layers and these layers have been shown in Figure above. The figure the hierarchy between these layers. The layers in the system are User Interface, Application, and Database. User Interface layer (which contains Display Module, Selection Module) is the layer at the top and User Interface layer is allowed to use Application and Database layer. Application layer (which contains Selection, Teacher Manager, Schedule Manager, Options Manager, and Database Manger) is placed as middle layer which do not have permission to use User Interface layer but have permission to use Database layer. Database layer (which contains Main Content DB, Main Account DB, and Backup DB) cannot use any of the layers because it is the layer at the bottom. User Interface, Application and Database layers provides cohesive a set of services that are semantically related. Each of these layers provides public interface facilities that maybe invoked or accessed by other software. Context diagram for layer view represents “is allowed to use” relation between the layers. Figure below demonstrates PRODERS' being developed on the top of the Network System Layer, Database Layer and Operating System Layer. PRODERS is allowed to use services of Network System Layer and Database Layer; while Network System Layer and Database Layer are allowed to use only services of Operating System Layer.

****

**9.5. ASPECTS VIEW**

PRODERS use aspect view to design a model for implementation of intersection troubles. In order to promote the modifiability by increasing modularity by avoiding the tangling of intersection functionality and business domain functionality we have used such view.



Aspect View of PRODERS

Figure above demonstrates an example about an Aspect view of PRODERS. Selection Management has an aspect type which is a specialized module that contains the implementation of intersected troubles in PRODERS. Selection Manager Aspect intersection calls to any operations in the Selection Manager package.

**9.6. DATA MODEL VIEW**

The Data Model Style is chosen to describe the static information structure of Proders project in terms of data entities and their relationships. The data model is represented graphically in entity relationship diagrams.

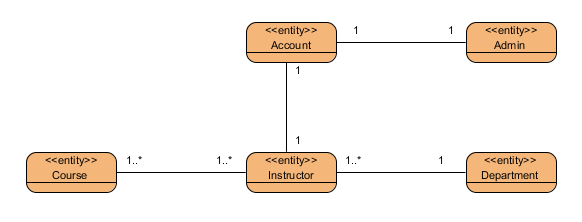


Figure 26

Figure 26 represents Data Model Style of Proders project. Data Model Style explains the relationship between data model, for this project the relationship between entities are as follows;

- One Admin can only have one Account, one headmaster

can just have one Account

- One or many headmaster can give one or many Course.

- One or many Student can attend one or many Course at the same time.

**10. COMPONENT AND CONNECTOR VIEWS**

Component and Connector View shows that how the architecture of a software system is structured as a set of elements that have run-time behavior and interactions. In of “Proders” project; Client-Server View, Publish-Subscribe View and Repository View are used in order to explain the run-time processes.

**10.1. Client-Server Style**

Client-server Style is used in Proders in order to show computational flow of the system. In case Client-Server Style provides a synchronous service invocation, this style is convenient for our project. In Client Server Style the requester of a service waits, until a requested service completes its actions, possibly providing a return result. The same procedure is applied in Proders project that client sends content-requests from server port to server (MainServer) and waits until getting services.



Figure 27

Figure 27 represents Elements of Client-Server Style of Proders. Client is a component that invokes services of a server component. Headmaster components are defined as a client typed components. Server is a component that provides services to client components. AccountServer and ContentServer are defined as a client typed components. Headmaster and AccountServer/ContentServer components include content-request/login-request-reply connectors to communicate with each other. This mechanism is used by a client to invoke services on a server. Request/reply connectors have two roles: a request role and a reply role. The relationship between Headmaster and AccountServer/ContentServer components are *attachment* relation, associates client service-request ports with the request role of the connector and server service-reply ports with the reply role of the connector.

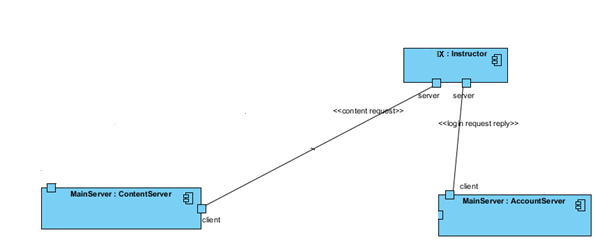


Figure 28

Figure 28 shows the client-server style of Proders. The Headmaster includes two client ports, which allows Headmaster to communicate and send requests to Headmaster and system components. Headmaster sends content requests to MainServer in order to get the course related contents such as course time, class, classroom, etc. from ContentServer. It also sends login request/reply messages to AccountServer in order to login to the system.

**10.2. Repository Style**

Repository Style is used in Proders in order to show which data accessors are connected to which data repositories. Components of this view are data accessors and repositories. In repository style, components are Content Server, Account Server, Main Content Database and Main Account Database.

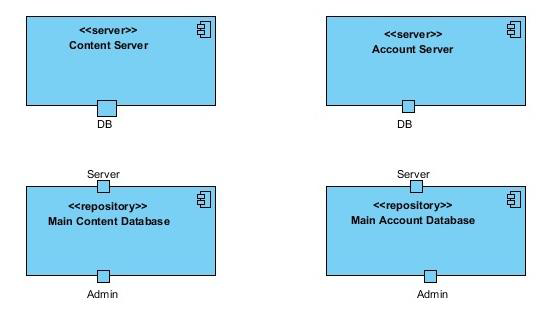


Figure 29

Figure 29 shows the two types of “server” and two “repository” types. Server types are Content Server, Account Server each has a DB port and repository types are Main Content Database and Main Account Database each has a Server and an Admin port.



Figure 30

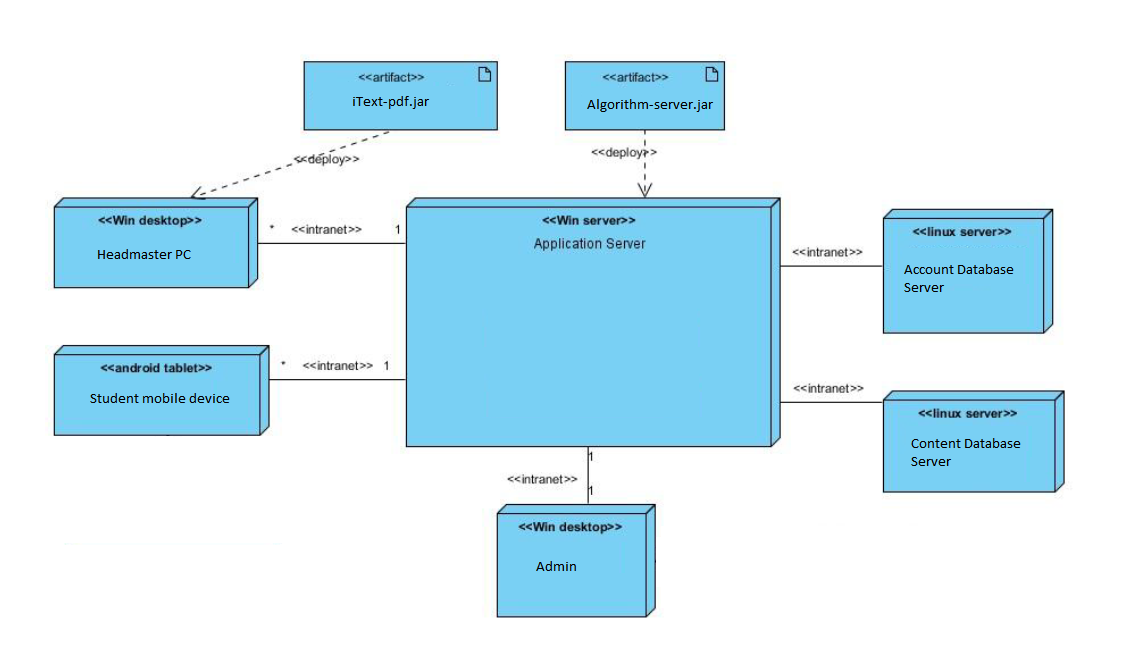
Repository View of the Proders has shown in the Figure 30. “Main” is an Account Server that accesses to Main Account Database called “DB”. Administrative connects to Main Account Database via DB port and also Administrative can connect to Main Content Database named DB Content via DB port. Content server accesses to DB Content with DB port.

**11. ALLOCATION VIEWS**

The allocation style presents a mapping of the architecture to its environment. Allocation views present a mapping from the elements of either a module or component and connector style onto elements of the environment. Relations allocated to relation is used to show which software element is allocated to which environmental element.

**11.1. Deployment View**

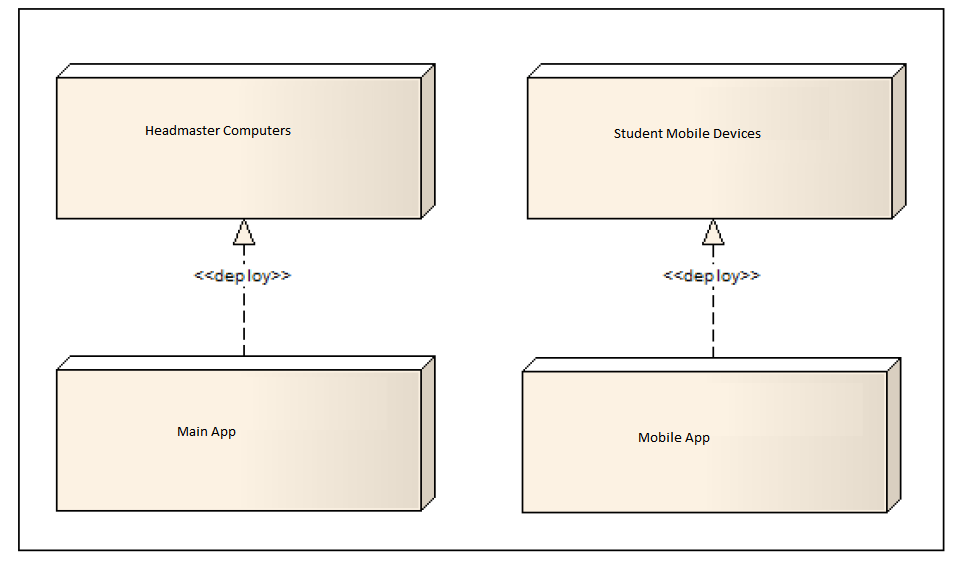
It shows the allocation relations of the software elements of ProDers system to their environment. Environmental elements shows software elements are used to deploy in deployment diagram. This deployment diagram serves to model the physical deployment of software artefacts on deployment targets.



-Deployment View

Deployment view of the system has shown in the above that shows the mapping between the software’s components and connectors and the hardware of the computing platform on which the software executes. The nodes in this view are Application Server which is a windows server, Headmaster Pc is Windows based computer, Student Mobile Devices which are android tablets, Admin which is the a Windows based computer for the Admin user, Account Database Server and Content Database Server are the Linux servers. These nodes are communicating with the main node which is the Application serve. The communication between Application Server and the other nodes is supplied with Intranet which is a network connects local area networks and wide area networks to each other. Headmaster PCs and Student Tablets can connect to Application Server via Intranet. Algorithm-server.jar an artifact that to load on the Headmaster PCs. Admin deploys the desktop computer of the Admin

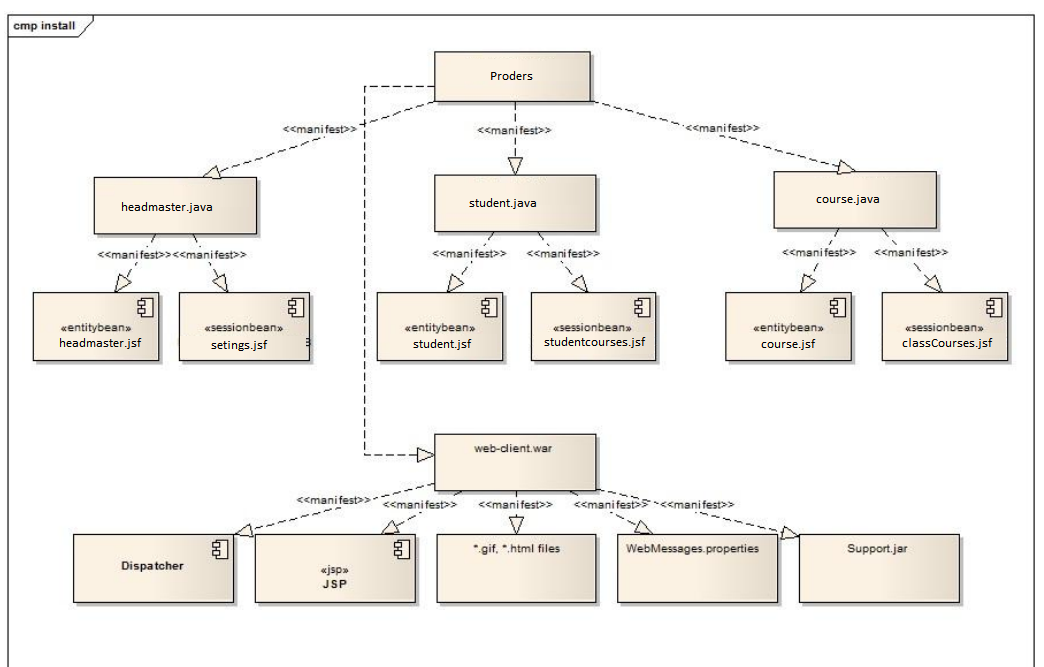
Context diagram for deployment view shown in below diagram which represents “deploy” relation between ProDers system and the PCs. The database of the system which 93 contains the account information and the lecture content is deployed to the servers. In addition, the software of the system is installed to the computers in the headmaster’s computers and also student’s mobile devices to run the ProDers Application.

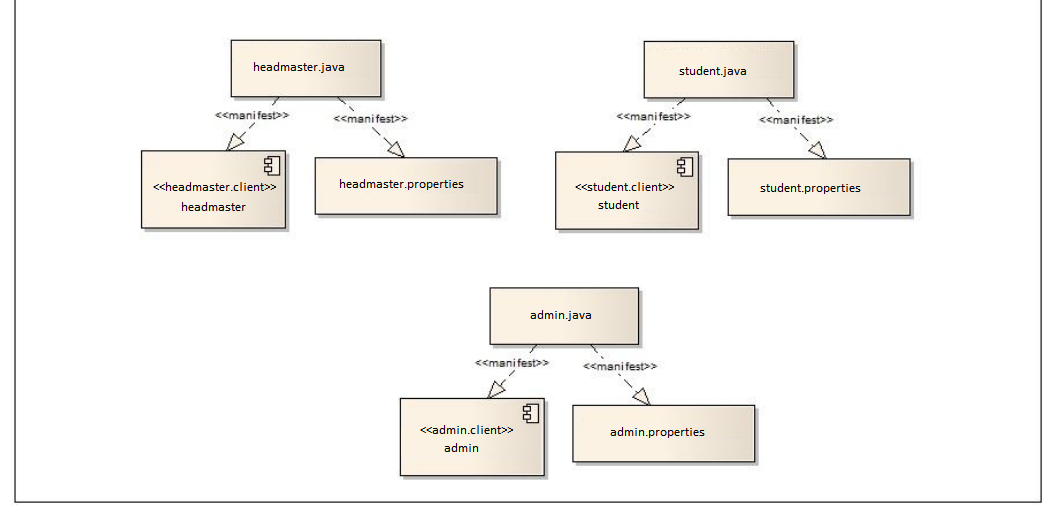


-Context diagram

**11.2. Install View**

It is the view of the system in production environment and it allocates the C&C components of the system to a file management system. Elements of the Install view except C&C components is the environmental elements such as files or directories. Install view is system to show how the installed system is organized as a structure of files and folders and describes how software elements map to that structure.



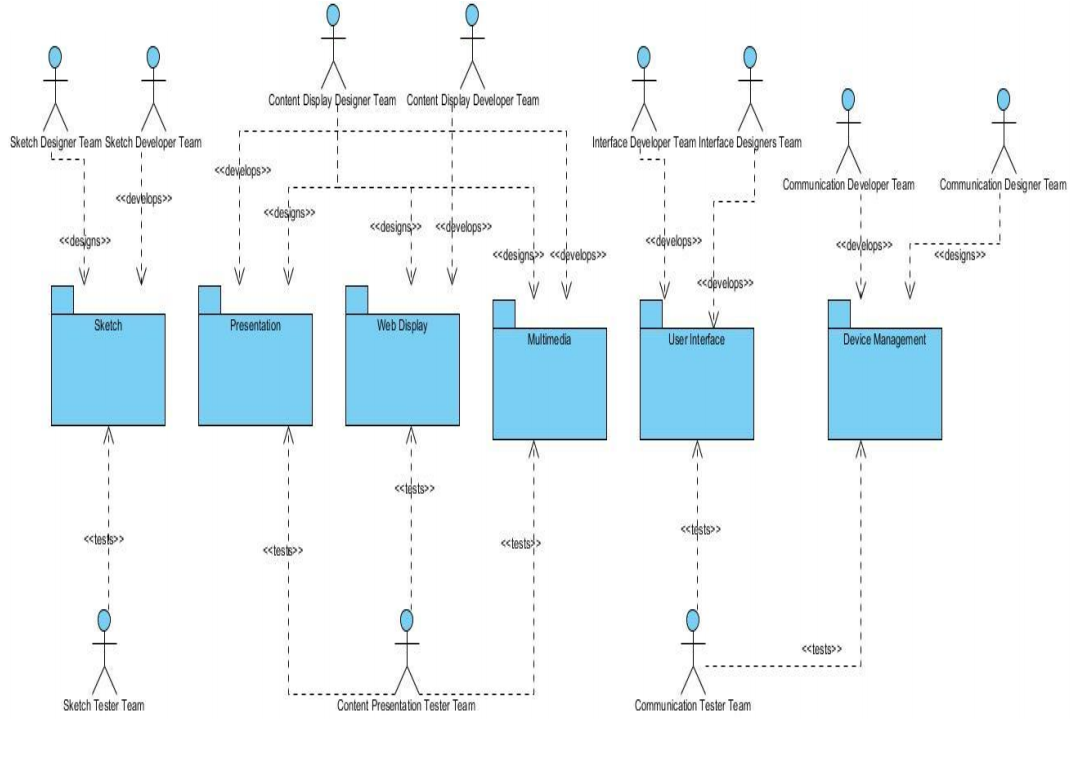


-Install View

Above figure shows the Install view of the ProDers App. Install view shows how the installed system is organized in ProDers. The software is going to be deployed to the Application Server is called “ProDers” which is a package consisting of file. First java application inside the enterprise archive ProDers is “headmaster.java” which is an application for Headmasters when they are logged into the application. It consists of two components headmaster.jsf and settings.jsf. headmaster.jsf is an entity bean which is used for persistent data storage for headmasters in the app. settings.jsf is a session bean which is used when it needs to change settings. Student.java is a java application in the app when students are using the app like if headmaster determined the curriculum and show the courses in detail. This java application consists of an entitybean called student.jsf and a sessionbean called studentcourses.jsf. student.jsf is an entity bean which is used for persistent data storage for students in the app. studentcourses.jsf is a session bean which is used when only one student has access to the app. Third java application is course.java which is to control and hold the course contents in the app. This java application consists of an entitybean called course.jsf and a sessionbean called classCourse.jsf. course.jsf is an entity bean which is used for persistent data storage for course contents in the app. classCourse.jsf is a session bean which is used to control course content. The Web archive file web-client.war is used to add web services to the system. This archive consists of dispatcher, Java Server Pages JSP, \*.gif, \*.html files, and a java application to support web services Support.jar. Proders.headmaster.java is the java application to be installed into the computer of the headmasters. It consists of headmaster.properties and headmaster component which is a headmaster.client. Proders.headmaster.java is the java application to be installed into the mobile devices of the students. It consists of students.properties and Student component which is a student.client. Proders.admin.java is the java application to be installed into the computer of the administrative. It consists of admin.properties and Admin component which is admin.client.

**11.3. Work Allocation View**

It is used in the app to describe the allocation of a module to a configuration item. Items are teams which are for developing, design and testing the system. Work Allocation view is to show the major units of the system that must be present to form a working system and who will produce them. Developing teams are for sketching, content display, interface, communication. Designer Teams are Sketch Designer Team, Content Display Designer Team, Interface Designer Team, and Communication Designer Team. Testing teams are Sketch, Content Presentation, Communication tester teams. In this view software elements are modules and their relations between configuration items are showed. There are three teams for testing and four teams for developing and design the system. Figure below shows each team and major units of the system as packages. Sketch module is the part of the system that is related with sketching and this module is designed by Sketch Designer Team, developed by Sketch Developer Team, and tested by Sketch Tester Team. Presentation module is one of the most frequently used parts of the system that is used in the lectures for presentations by Headmaster. Presentation module is designed by Content Display Designer Team, developed by Content Display Developer Team, and tested by Content Display Tester Team. Web Display module is a unit of the system for displaying web contents and connecting to the internet. This part is designed by Content Display Designer Team, developed by Content Display Developer Team, and tested by Content Display Tester Team. Multimedia module is to play videos for lectures. Multimedia part is designed by Content Display Designer Team, developed by Content Display Developer Team, and tested by Content Display Tester Team. User Interface module of the system is for designing and developing all the interfaces of proders that user is facing with. User Interfaces are designed by User Interface Design Team, developed by User Interface Developer Team, and tested by Communication Tester Team. Device management is the module that makes the system connected to a local network to be connected to Headmaster ‟ computer, admin‟s computer and students‟ tablets. Also device management module should provide the device with connection to the internet. This part is designed by Communication Designer Team, developed by Communication Developer Team, and tested by Communication Tester Team.



-Work Allocation View

**12. EVALUATION OF THE SOFTWARE ARCHITECTURE**

Software architecture is considered to be one of the most important design tools to determine whether a system is in pre-design, active deployment or maintenance to understand a software system. The evaluation of the software architecture is necessary to analyze the system quality at the beginning of the life cycle and to predict the quality of the system before it is installed. The assessment provides a mechanism to identify the potential risks and how the system will evolve. To do this, we evaluated the producers with scenario-based architectural analysis (SAAM) and ATAM.

12.1. Scenario Based Architectural Analysis (SAAM)

The Proders has a wide and complex scope, and it is difficult to understand the architecture from the gross level to the detailed pieces. Due to the complexity of the project, the software architecture is evaluated step by step. The first step is to determine what the stakeholders are and what the expectations are from the Proders project. The second step is to investigate all requirements with use cases. The main components of the project and the extension of the scope are defined according to these use cases. General architecture is defined as a result of these main steps. General architecture is defined as a result of sequential recurring processes. However, some problems and missing concepts may arise in the architecture of a complex project like Proders; It is decided to make an improvement on the architecture of the Proders project. At the evaluation stage, scenario-based assessment is chosen as the most effective way to analyze the qualities of the system in the early stages of the life cycle and to predict the qualifications of the Authorized Person.

12.1.1. Description of Candidate Architecture

The whole report before the section 12 includes the architecture of the Proders system.

12.1.2. Developed and Prioritized Scenarios

Proders is a huge project which has a number of stakeholders. The most important stakeholders who are mainly interested in the software architecture of Proders system are headmaster & student (customers), administrator, software/hardware engineer, network administrator and user interface designer. Therefore, the system should respond the needs of these stakeholders and should capture all the relevant scenarios. Below there are scenarios which are relevant with each stakeholder and arranged according to their priorities.

\*  Headmaster:

-  Login to system

-  Add/delete course

-  Change course

-  Change identity information

- Change access permission to the system

\*  Student:

-  Access schedule content

-  Change identity information

\*  Software Engineer:

-  Add a new feature to the system.

-  Port to another operating system

\* Network Administrator:  
- Change the network system

\* User Interface Designer:  
- Make modifications in user interface

100

12.1.3. Scenario Evaluation

The table shown in below indicates the priority of the scenarios mentioned in section 12.1.2, their impacts on the architecture of the system and required changes to implement some scenarios.

1. Login to system
2. Add/delete course
3. Change course
4. Change identity information
5. Change access permissions

12.1.4. Scenario Interactions

For the indirect scenarios, the corresponding changes required in the system are defined. Then, the number of changes for each component which requires modifications are identified and a table indicating this relation is constructed (Table 23).

1. Application module
2. Presentation manager module
3. Display module

12.1.5. Overall Evaluations

At the end of the evaluation phase, we have defined the scenario interactions and identify bad and good scenarios. We want the architecture that we defined for Proders provide high cohesion and low coupling. However, some of the scenarios affect the same component, especially Application Module. In order to prevent this, Application Layer could be separated into smaller layers. Apart from those, there are not any semantically unrelated scenarios which affect the same architectural component which is also illustrated in table 23. As a result most of the scenarios that require change correspond to only one component of the system, which appears to prove the validity of software architecture of the Proders. Minor changes in the architecture would be adequate to create high- quality software architecture of the Proders.

12.2. UTILITY TREE (ATAM)

ATAM is used to assess the results of architectural decisions based on multiple quality requirements and at the same time to understand how these quality attributes interact with each other. ATAM helps to understand how the quality features of the employer system are gaining each other. The ATAM participants of the Proders system will be evaluation team, project decision makers and architectural stakeholders.

**13. CONCLUSION**

This report is designed to design qualified software architecture for Authorized Dealers. In the design phase of the system, the first step was requirement analysis. Stakeholders and system requirements and expectations are determined. Project analysis, usage diagrams, architectural scenarios and prototypes have been strengthened. Then the technical problems identified according to needs are analyzed. The next step is to analyze the solution area that maps areas of each technical problem. Information sources and derived concepts are defined. The description of the software architecture process of the project continues to find appropriate styles from the Modeller software architecture Views and Beyond Approach. In Module Styles; The view, the parsing view, the generalization view, the layer view, the view view, and the data flow views are used to document application units of a Generator's basic units. Component and Connector Styles; The client-server view, publish-subscribe view, and repository views are implemented to show how the system's architecture is structured as a set of elements with run-time behaviors and interactions. The architectural design process has been successfully completed to fulfill the purpose of the Proders project. In Allocation Styles; Deployment view, setup view, and job distribution views are implemented to match the system's software architecture to the environment. Finally, software architectural evaluation processes, such as ATAM, are implemented to understand the missing and incorrect parts of the Architect's architecture.

In the technical problem analysis section, we experienced some difficulties when describing the problems. However, we were able to improve our creativity and increase our knowledge of technical problem analysis. We learned how to determine the mathematics and application area, computer science and quality problems. Moreover, when conducting area analysis, we searched for the solution areas that we determined according to technical problems. We spent a lot of time exploring the domain names from various sources from the internet and books. While choosing the right views for the application, we faced some problems with the modeling phase of the software architecture. However, we were able to increase our creativity by investigating and studying similar projects to increase our knowledge of architectural appearance and modeling software architecture. Finally, we searched all the details in detail, then selected the appropriate views of the In-App and Beyond Approach, then easily apply them to our software architecture.

We gained a lot of experience and we had the chance to practice our knowledge while performing this project. As a result, this project has met some obstacles, but we have taken care of everything and have increased our knowledge of software architecture design steps. In the future, according to the information we receive from the software architecture evaluation process, we will monitor our architecture; We need to add new modules to our system and improve accordingly. In general, we fulfill high-quality architectural requirements and have implemented a successful architecture for the Proders project.