<Project Name>

Architecture Notebook

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Purpose

Purpose of Architecture Notebook is collecting all properties about system design and standards in one file. These properties include architecture goals, philosophy, assumptions and dependencies, the requirement of design, architectural mechanism, design constraints, justifications, and decisions. At this point, we can easily check whether we follow the rules or not the project after each step. Besides, we can have a standard on the Architectural side of the project. So, we can easily say this document concretize the properties of the system and it helps to the new team member for adapting to the project.

# Architectural goals and philosophy

In this project, we aim to design sports center’s mobile and web application. At this point, our project is divided into two different sides. These are member and manager sides. In member side, we can have a wide variety of user profiles. So, the first architectural goal is simplicity. It is important to us that users can use the application easily. The members may want to connect with many different devices. Therefore, we need to provide various device support. (For this title, we decide to use Android version 4 or newer). In the mobile application, we aimed to keep the size of the mobile application and system requirements as less as possible. So, we can reach maximum population of members. In management side, we need to share statistics of centers clearly with managers. A misunderstanding at this topic can cause bad sequences for the firm. Our system doesn’t need any additional hardware specializations. In website, we work on modern web browser. But, we plan to increase browser support.

# Assumptions and dependencies

In this project, we will develop web and android application for sports center management. We use Angular5 for web application side because of javascript languages are easier than other languages which are using for web application’s implementation. Also, our team members have no knowledge about web designing and implementing so, we have to learn fast and pass on implementing.

We use and android because of implementing an android application’s implementation is easier than other mobile platform applications, android is widely used and our team members have knowledge about it.

We use MySQL for database side, because of easier and simpler than other database services and our team have knowledge about it. Also, our server supports MySQL and server processes about database became easier.

We use java for control database, web and android applications’ base. Firstly, we have to use and object-oriented language like c# or java. Secondly, we are more experienced about java programming. Also, java community is better than c# community, any problem’s solution is easier to find for java.

# Architecturally significant requirements

### Why Use the Spring Framework?

<http://www.wrox.com/WileyCDA/Section/Why-Use-the-Spring-Framework-.id-130098.html>

### Why JavaScript Is and Will Continue to Be the First Choice of Programmers?

<https://dzone.com/articles/why-javascript-and-will>

### Why is the Android OS So Popular?

<http://opensourceforu.com/2016/02/why-is-the-android-os-so-popular/>

### Why MySQL is still king?

<https://www.infoworld.com/article/3195764/nosql/nosql-no-problem-why-mysql-is-still-king.html>

[Insert a reference or link to the requirements that must be implemented to realize the architecture.]

NEDEN OOP KULLANDIK NEDEN SPRİNG KULLANDIK MAKALEYLE AÇIKLA

# Decisions, constraints, and justifications xxxx

[List the decisions that have been made regarding architectural approaches and the constraints being placed on the way that the developers build the system. These will serve as guidelines for defining architecturally significant parts of the system. Justify each decision or constraint so that developers understand the importance of building the system according to the context created by those decisions and constraints. This may include a list of DOs and DON’Ts to guide the developers in building the system.]

* Decision or constraint and justification
* Decision or constraint and justification

# Architectural Mechanisms

[List the architectural mechanisms and describe the current state of each one. Initially, each mechanism may be only name and a brief description. They will evolve until the mechanism is a collaboration or pattern that can be directly applied to some aspect of the design.]

## Architectural Mechanism 1

[Describe the purpose, attributes, and function of the architectural mechanism.]

## Architectural Mechanism 2

[Describe the purpose, attributes, and function of the architectural mechanism.]

# Key abstractions

[List and briefly describe the key abstractions of the system. This should be a relatively short list of the critical concepts that define the system. The key abstractions will usually translate to the initial analysis classes and important patterns.]

# Layers or architectural framework

[Describe the architectural pattern that you will use or how the architecture will be consistent and uniform. This could be a simple reference to an existing or well-known architectural pattern, such as the Layer framework, a reference to a high-level model of the framework, or a description of how the major system components should be put together.]

# Architectural views

[Describe the architectural views that you will use to describe the software architecture. This illustrates the different perspectives that you will make available to review and to document architectural decisions.]

## Recommended views

* **Logical:** Describes the structure and behavior of architecturally significant portions of the system. This might include the package structure, critical interfaces, important classes and subsystems, and the relationships between these elements. It also includes physical and logical views of persistent data, if persistence will be built into the system. This is a documented subset of the design.
* **Operational:** Describes the physical nodes of the system and the processes, threads, and components that run on those physical nodes. This view isn’t necessary if the system runs in a single process and thread.
* **Use case:** A list or diagram of the use cases that contain architecturally significant requirements.