Bilkent University

Department of Computer Engineering

# Senior Design Project

*BFSP*

## High-Level Design Report

Rıdvan Çelik, Solehjon Ruziboev, Ulugbek Irmatov

Supervisor: Uğur Doğrusöz

Jury Members: Fazlı Can and Çiğdem Gündüz Demir

Dec 22, 2017

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

# 1. Introduction

In this Report, purpose of the system will be briefly mentioned again. Then Software Architecture and Subsystem Services will be explained in detail.

## 1.1 Purpose of the System

BFSP will provide students with all in one platform which is a solution for the problems mentioned in the Analysis Report. It consist of question & answer, teacher rating according to some features of instructors, course material trading, tutorship and online course resources pages. Additionally, there will be announcement page that can display club events etc.

Target audience of the project is all students at Bilkent University. Actually, everyone will able to access the web page, however only Bilkent students may have an account. The use of the web page will be limited for visitors. The difference between member and guest had been explained in the Analysis Report.

## 1.2 Design Goals

Maintainability, user-friendliness, and performance are main design goals. Since we hope that the system will be used for at least several decades, it should be easy to maintain for developers. Simplicity is important, the information should be displayed to users in the simplest way possible. It shouldn’t be confusing, too colorful, or hard to get around the system. The system should have low response time. Especially, the search component of the system should be high-performing to show search suggestions promptly.

Design goals such as adaptability and portability are part of implementation. It should be portable because developers work on different development environments. Developers should be able to easily adapt the system to new requirements as well.

Design trade-offs are considered during implementation of the system. We will consider usability-utility and security-performance trade-offs.

### Usability - Utility

We plan to do our best to balance usefulness and ease of use. The system should have several useful functionalities for users. We will prefer ease of use over too many functionalities though. Since our main goal is to provide answers, we can leave out some secondary functionalities if they worsen usability.

### Security - Performance

We will prioritize security over performance when necessary. We still plan to make high-performing system but when we have to do extra operations to ensure security we will do it.

## 1.3 Definitions, Acronyms, and Abbreviations

| | |
|---|---|
| React.js | UI library developed by Facebook. |
| MongoDB | NoSQL database |
| Express | A Node.js framework |

## 1.4 Overview (of the system)

The system consists of three layers. In the first layer, there’s UI of they system implemented in React. In the second layer, there’s application logic of the system implemented in Node.js and Express.js. In the third layer, the system has MongoDB driver that manages all the data of the system. This architecture is shown in more detail in subsequent sections of this report.

# 2. Proposed Software Architecture

## 2.1 Overview

We will use 3 - tier architecture. First tier is frontend developed by React.js, second tier is application (middleware) tier that manages UI and database, and third tier is database driver that is responsible for data management.

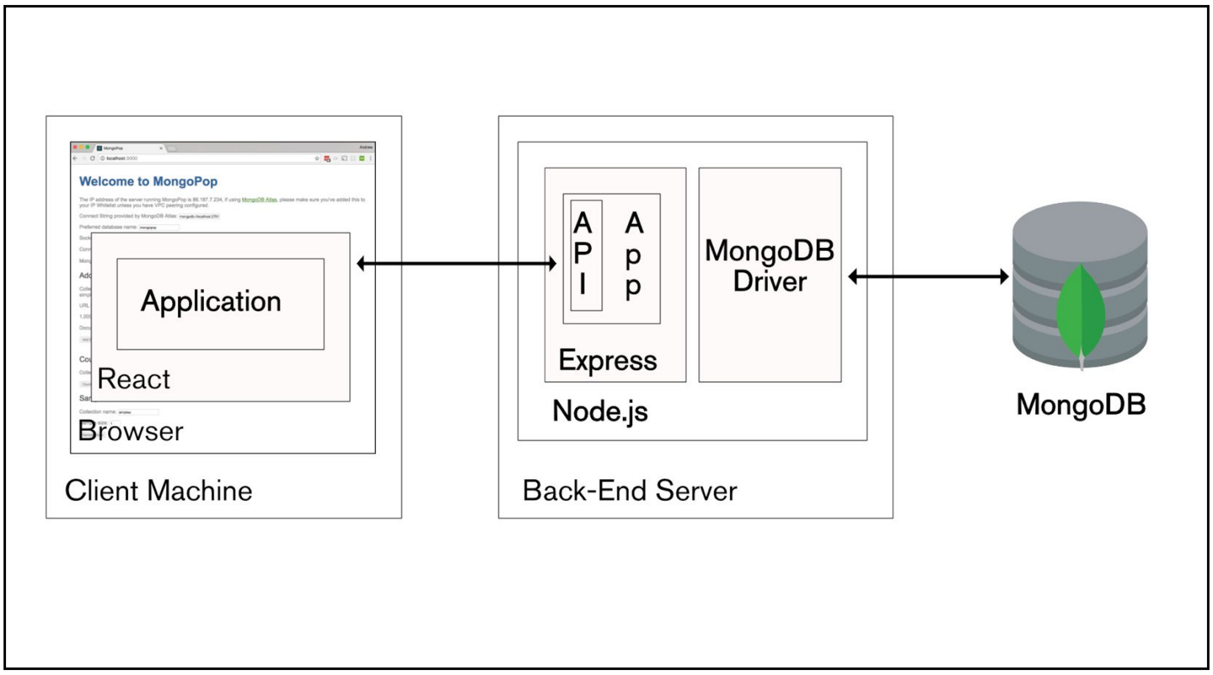


Figure 1 Overview of the system architecture [1]

## 2.2 Subsystem Decomposition

In the first tier, there is User Interface component. Second tier contains application logic components such as App management, Ad management, User management, Search management, Post management, and Course management. In the third tier, there is Data management component. These components' functions are explained in detail in subsystem service section.

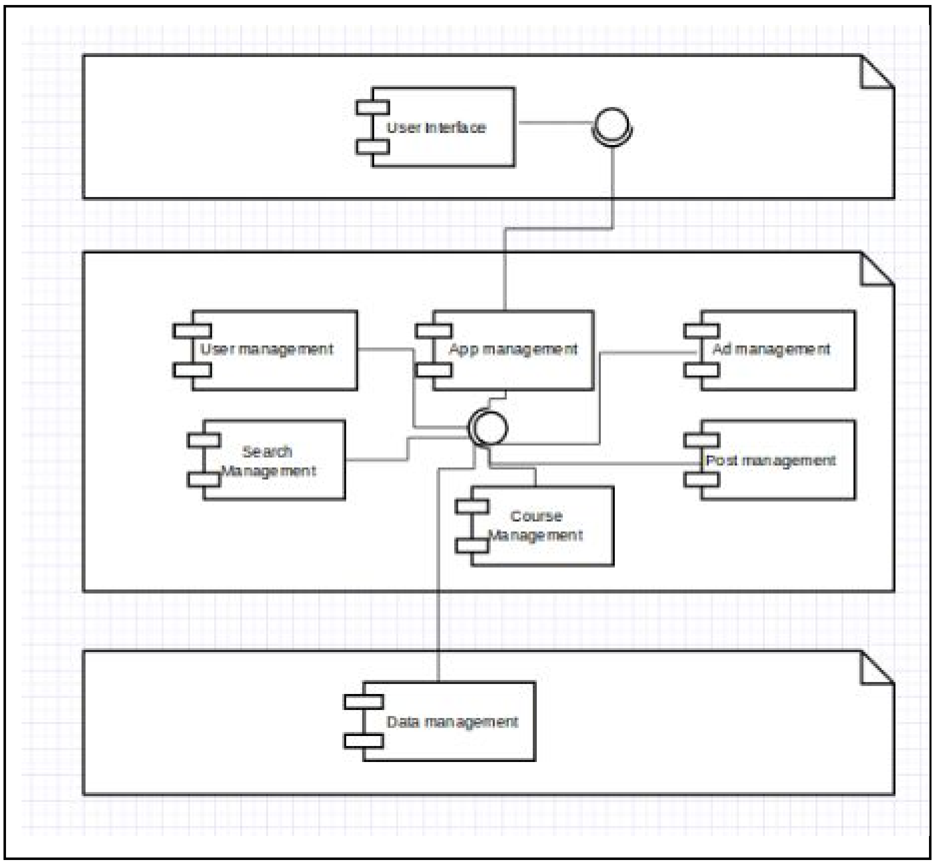
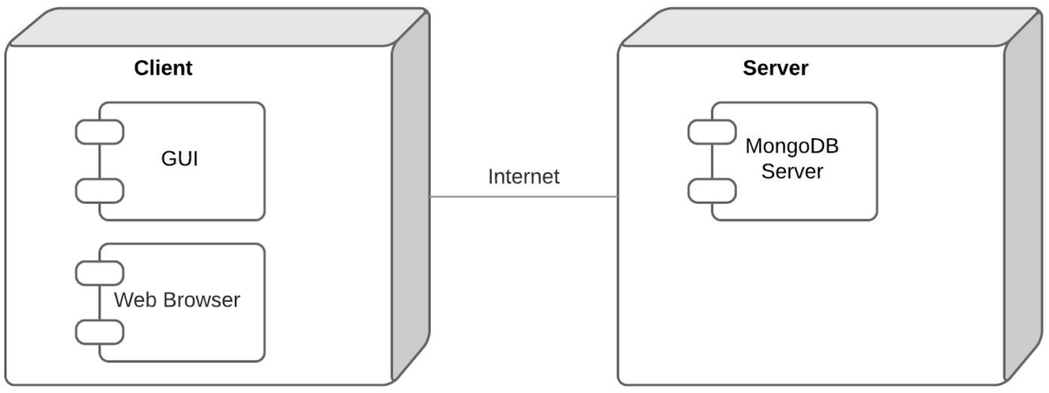


Figure 2 Subsystem decomposition of the system

## 2.3 Hardware/Software Mapping

Since the architecture of the system is client - server, we have client machine and server machine.  As shown in figure below, client makes request and server sends back requested data.



## 2.4 Persistent Data Management

MongoDB is more appropriate and compatible in order to make our web application's data persistent. As mentioned in previous sections, the system's subsystem responsible for data management is MongoDB driver. Since, we chose to develop our application in JavaScript. We decided to develop it in full-stack JavaScript so that we can use the same language in both backend and frontend. Modern JavaScript full-stack is MongoDB, Express, React, and Node.

## 2.5 Access Control and Security

The data and features of the website are going to be accessed either by a logged in user or a guest user. Also there is an administrator who controls the website and users. The logged in user will have a lot more options in the website rather than a guest user. A guest user can only view the contents of the web page without being able to add any data. The logged in user however can use pretty much every available feature such answering and posting new questions, rating teachers, adding course materials or online resources, buying available materials or post or hire a tutor. The access to the data is going to be split between these users in a natural way, in other words in the way that is logical for both users to have an availability of features and data. The administrator on the other hand has an access to all data of the webpage. Additionally, he can control which information should be hidden, for instance an inappropriate comments about a teacher. Also an admin can block users whose actions were not following the ‘social’ rules of the web page. The administrator additionally controls which announcements should be shown in the main page.

## 2.6 Global Software Control

Initialization of the requests and subsystem synchronization have been mentioned in other sections in detail. Since our application does not require much complexity on hardware level, it won’t be discourse about global software control. Concurrency and  synchronization are remissible for now [2].

## 2.7 Boundary Conditions

- Initialization - General flow of events occur during initial web page loading. The process is the same as in any web application.
- Failure - Internet connection may be timed out. So, we should prepare the system for this situation. The system will keep displaying the page instead of error page at the time of internet connection timeout. Connection timeout page is shown only if user clicks for any functionality. This may be done by storing all the data locally on browser. If this is done by browser itself, then they system should ensure that the data of timeout time is saved. So that the user can continue from where it was left off.
- Termination - The system terminates when browser window or web application’s tab is closed.

# 3. Subsystem Services

- UI management - This component is responsible for frontend : showing data and getting input from user.
- App management - This is main component that includes and connects all other components.
- Post management - This component manages questions and answers. When user submits question or answer, it will make sure they are saved in the database. Also, it manages upvotes and comments of question/answer post.

- User management - This component manages authentication and authorization in the system. It checks if user is authorized to perform post, comment, upvote, and rate operations. Only logged in user can perform those operations.
- Search management - This component must respond to user’s input in search by showing suggestions.
- Ad management - This component is responsible for displaying announcements or advertisements on the page. When user clicks on ad, it must open ad link on new tab.
- Course management - This component is responsible for filtering posts according to courses. When user selects “CS491”, only question and answer related to “CS491” are displayed.
- Data management - This component is MongoDB driver responsible for managing all the data stored in database.

## 4. Glossary

MERN stack - Modern JavaScript full stack for web development (MongoDB, Express, React, Node)

## 5. References

1. MongoDB. (2017). *The Modern Application Stack - Part 5: Using ReactJS, ES6 & JSX to Build a UI (the rise of MERN)*. [online] Available at: https://www.mongodb.com/blog/post/the-modern-application-stack-part-5-using-reactjs-es6-and-jsx-to-build-a-ui-the-rise-of-mern [Accessed 16 Dec. 2017].
2. “System Design Document Template,” *RSS*. [Online]. Available: http://www.utdallas.edu/~chung/SP/SystemDesignDocumentTemplate.htm. [Accessed: 22-Dec-2017].