**WizeStream**

Damir Bar,

Dima Volgin,

Eran Sterman,

Shay Cohen,

Sefi Shalom

Table of Contents

1. Introduction 3

1.1 Abstract 3

1.2 Related Work 3

1.3 Scope 5

1.4 Definitions, Acronyms and Abbreviations 7

1.5 Technologies to be used 8

1.6 Overview 9

2. Overall Description 10

2.1 Product Perspective 10

2.2 Data Dictionary 11

2.3 ER Diagram -

3. Specific Requirements 16

3.1.1 Use-Case Reports -

3.1.2 Class Diagram -

3.1.3 Supplementary Requirements -

3.2 Functional Requirements 16

3.2.1 User Class 1 - The Student 16

3.2.2 User Class 2 - The Teacher 20

3.2.3 User Class 3 - The University -

1. **Introduction**

## Abstract

Over the past hundred years, the educational system never evolved. By the 00’ it improved massively by using a wide range of diﬀerent services and applications to ease the learning experience, such as:

* WhatsApp, Facebook - Forum/messaging tools for group chatting, messy ﬁle sharing system and QA.
* Google Drive, Dropbox - Generic not student-oriented cloud system. Old communication system between students and lecturers, an exercise submission feature and QA option.
* Gool - Non-free on-demand video courses.
* Kahoot - Interactive student teacher app.
* Institution based feedback. wizeStream is a reality which combines and enhances these features and more in to one easy-to-use user-friendly platform.

## Related Work

This chapter surveys previous student-teacher platforms. Our project isn’t about reinventing the wheel, but to suggest a decent way to bind between existing products and to innovate with our live streaming.

We’ll demonstrate for each field how it exists, but suffers lack of functionality and accessibility for students/teachers.

1. **Clouds**

As we all know, clouds exist for a very long time. There are clouds owned by the strongest companies in the world. Our project isn’t meant to compete with any of those, but to optimize the student/teacher approach to a study-relevant cloud. Our application will identify the file’s extension, decide whether it’s a document, a summary or a previous test, and will sort it so the user will enjoy surfing our cloud, instead of getting lost in it.

1. **Text-Message systems**

There are lots of text-messaging systems, such as WhatsApp, Telegram, etc. Our platform will suggest a basic messaging system.

1. **GPA Calculator**

During our studies, we came across several GPA calculators. One of them was in our university’s mobile application and the other was an external mobile application. The problem with our university’s application was the fact that it was unmodifiable. We couldn’t calculate our “what ifs” or to check whether we should retake a certain course. The problem with the external application was the fact that it’s external. It forced us to download a third-party application to our mobile phones.

1. **Tracking course progress**

Our product provides students with a course progress tracking feature, based on taking quizzes, submitting assignments and reading papers. Our team has yet to encounter such a feature.

1. **Learning management system**

There are many learning management systems. The one which the project’s team used is Moodle. Moodle suggest a lot, but as you will witness, we suggest more.

1. **Live Stream Video**

This is where we truly innovate. Using the product of our project, you will be able to be at the lecture, even if you are absent from the class. You can send/receive messages from other students from the same lecture group during the lecture. You can ask the group questions and answer questions yourself.

1. **Live Feedback**

Everyone benefits from the Live Feedback. Students which haven’t quite gotten the last topic learned in class, will no longer desperately raise their hands, but post it right into our Live Feedback system in real time. If most of the students felt the same way, the teacher might want to consider it. If a student did understand, he can tell the application that he did understand and by so to balance the “wizeScale”.

1. **Statistical Gathering**

The product of this project will independently gather statistical information regarding students’ satisfaction and students’ progress. This way the educational institution will be able to monitor its employees and improve.

**References:**

* “Using Moodle” – Jason Cole & Helen Foster - <https://books.google.co.il/books?hl=iw&lr=&id=wfPPb1m0G6EC&oi=fnd&pg=PR5&dq=moodle&ots=vSxkIj1BMB&sig=KNtDANrMBc9EjjWFI_mHWiQiFvM&redir_esc=y#v=onepage&q=moodle&f=false>
* “Moodle” – Klaus Brandl - <http://llt.msu.edu/vol9num2/review1/default.html>
* “Changing How and What Children Learn in School with Computer-Based Technologies” - Jeremy Roschelle, Roy D. Pea, Christopher M. Hoadley, Douglas N. Gordin & Barbara Means - <https://telearn.archives-ouvertes.fr/hal-00190610/document>

## Scope

* Create different users with varied roles and scopes.
* User Experience:
  + Live stream video
  + Tracking course progress
  + wizeStream’s annual extraordinary
* User convenience
  + Grades Average calculator
  + File system
  + Q&A
  + Student answer providing instead of teacher
  + Android/IOS/Web UI
  + Messaging system
* Statistical collection
  + Live feedback
  + Fame system
  + Scoring system
* Security
  + Spamming/Abusive user reporting system
  + Anonymity option
  + General security
* Accessibility
  + Multiple language support

## Definitions, Acronyms and Abbreviation

**HTML** – Hyper Text Markup Language.

**HTTP** – Hypertext Transfer Protocol.

**Doc** – document.

**MongoDB** – Document-oriented database program.

**JavaScript** – Programming language

**Node**.js – JavaScript run-time environment for executing JavaScript code server-side.

**IDE** – Integrated Development Environment. A software application which provides comprehensive facilities to computer programmers for software development.

**Android Studio** – The official IDE for Google’s Android operating system.

**WebStorm** – An IDE primarily for web development.

**AngularJS** – JavaScript front-end web application framework.

## Technologies to be used

# Server API - Node.js

# Database Application - MongoDB

# Development tool - Android Studio and WebStorm

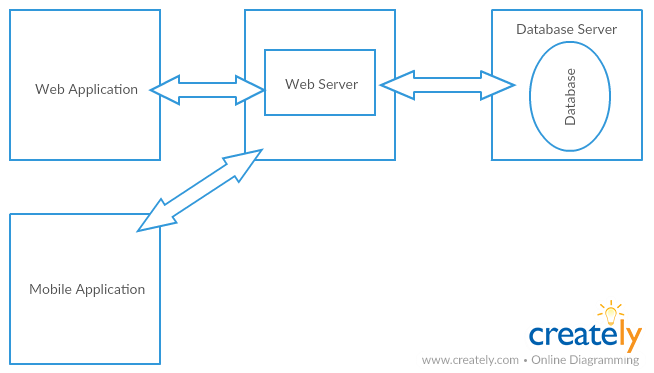
# Front-End framework - AngularJS

## Overview

wizeStream is a tool to help students get better at their studies and to help teachers have a better perspective on their class. wizeStream allows teachers to upload relevant ﬁles, answer students’ questions and improve their teaching methods using wizeStream feedback. With wizeStream students can communicate with their classmates, share ﬁles and enjoy live video streaming, providing the option of live feedback. wizeStream is the best way to gather statistical information for the use of both parties, students and teachers, as well as any department chosen. wizeStream will drive both sides, students and teachers, to thrive using a scoring system. wizeStream will make teaching and learning easier, faster and better.

# Overall Description

## Product Perspective



## Data Dictionary:

1. Student

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Unique** |
| ID | Long | True |
| First name | String | False |
| Last name | String | False |
| Display Name | String | False |
| Email | String | True |
| About me | String | False |
| Facebook ID | Long | True |
| Country | String | False |
| City | String | False |
| Age | Integer | False |
| University | Object – University | False |
| Gender | String | False |
| Courses | Object Array – Course | False |
| Photos | Array | False |
| File System | Object Array - references | True |
| Grades | Object Array – Grade | False |
| Credit | Long | False |
| Fame | Long | False |
| Messages | Object Array – Message | True |
| Register date | Date | False |
| Last updated | Date | False |
| Notifications | Object Array - Notification | True |

1. Teacher

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Unique** |
| ID | Long | True |
| First name | String | False |
| Last name | String | False |
| Display Name | String | False |
| Email | String | True |
| About me | String | False |
| Facebook ID | Long | True |
| Country | String | False |
| City | String | False |
| Age | Integer | False |
| University | Object – University | False |
| Gender | String | False |
| Courses | Object Array – Course | False |
| Photos | Array | False |
| File System | Object Array - references | True |
| Grades | Object Array – Grade | False |
| Credit | Long | False |
| Fame | Long | False |
| Messages | Object Array – Message | True |
| Register date | Date | False |
| Last updated | Date | False |
| Notifications | Object Array - Notification | True |

1. University:

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Unique** |
| ID | Long | True |
| Name | String | False |
| Teachers | Object Array - Teacher | False |
| Departments | Object Array - Department | False |
| Phones | String Array | True |

1. Department:

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Unique** |
| ID | Long | True |
| Name | String | False |
| Head | Teacher | False |
| Courses | Object Array - Department | False |
| Teachers | Object Array – Teacher | False |

1. Course:

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Unique** |
| Course number | Long | True |
| Name | String | False |
| Teacher | Teacher | False |
| Students | Object Array – Student | False |
| File System | Object Array - references | True |
| Location | String | False |
| Q&A | Object Array | True |
| Quiz | Object Array | True |
| Test | Test | True |
| Exercises | Object Array | True |
| Points | Integer | False |
| Messages | Object Array – Message | True |
| Register date | Date | False |
| Last updated | Date | False |

1. Test:

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Unique** |
| ID | Long | True |
| Name | String | False |
| Teacher | Teacher | False |
| Students | Object Array – Student | False |
| Location | String | False |
| Points | Integer | False |
| Dates | Date Array (2 elements) | False |

# Specific Requirements

**3.1.1 Use-Case Reports**

**3.1.1 Class Diagram**

**3.1.1 Supplementary Requirements**

**3.2 Functional Requirements**

**3.2.1 User Class 1 – The Student**

**3.2.1.1 Functional requirement 1.1**

**ID: FR1**

TITLE: Download the mobile application

DESC: A student should be able to download the mobile application through either an application store or similar service on the mobile phone. The application should be free of charge.

RAT: For a student to download the mobile application.

DEP: None

**3.2.1.2 Functional requirement 1.2**

**ID: FR2**

TITLE: Download and notify students of new releases

DESC: When a new/updated version or release of the software is released, the student should check for these manually. The download of the new release should be done through the mobile phone in the same way as downloading the mobile application.

RAT: For a student to download the most updated release.

DEP: FR1

**3.2.1.3 Functional requirement 1.3**

**ID: FR3**

TITLE: Student registration – Mobile and Web application

DESC: Given that a student has downloaded the mobile application or has browsed to the platform’s homepage, then the student should be able to register through the mobile application or the website. The student must provide first name, last name, password, e-mail address and university.

RAT: For a student to register on the mobile application or the website.

DEP: FR1 for Mobile application, None for Web application.

**3.2.1.4 Functional requirement 1.4**

**ID: FR4**

TITLE: Student log-in – Mobile and web application

DESC: Given that a student has registered, then the student should be able to log into the application. The student’s log-in information will be stored on the phone or browser, and in the future the student should be logged in automatically.

RAT: For a student to be logged-in after registration.

DEP: FR1, FR3

**3.2.1.5 Functional requirement 1.5**

**ID: FR5**

TITLE: Retrieve password

DESC: Given that a student has registered, then the student should be able to retrieve his/her password by e-mail.

RAT: For a student to retrieve his/her password.

DEP: FR4

**3.2.1.6 Functional requirement 1.6**

**ID: FR6**

TITLE: Mobile/Web application - Homepage

DESC: Given that a student is logged in to the application, then the first page that is shown should be the homepage, with the student’s profile picture on the navigation bar.

RAT: For a student to see his/her personal homepage.

DEP: FR4

**3.2.1.7 Functional requirement 1.7**

**ID: FR7**

TITLE: Personal profile page

DESC: Given that a student is logged in, then the student should be able to view his/her personal profile page.

RAT: For a student to see his/her personal profile page.

DEP: FR4

**3.2.1.8 Functional requirement 1.8**

**ID: FR8**

TITLE: Personal course page

DESC: Given that a student is logged in, then the student should be able to view his/her personal course page.

RAT: For a student to see his/her personal course page.

DEP: FR4

**3.2.1.9 Functional requirement 1.9**

**ID: FR9**

TITLE: Free-text search

DESC: Given that a student is logged in, then the student should be able to conduct a search by providing either course name, first name, last name or university name. The results are displayed in a list view by default.

RAT: For a student to search through the free-text search.

DEP: FR4

**3.2.1.10 Functional requirement 1.10**

**ID: FR10**

TITLE: Sorting results

DESC: Given that a student has conducted a search, the student should be able to sort the results according to the result type: teacher, student, course or university.

RAT: For a student to sort his/her search.

DEP: FR9

**3.2.1.11 Functional requirement 1.11**

**ID: FR11**

TITLE: Filtering results

DESC: Given that a student has conducted a search, the student should be able to filter the results according to the result type: teacher, student, course or university.

RAT: For a student to filter his/her search.

DEP: FR9

**3.2.1.12 Functional requirement 1.12**

**ID: FR12**

TITLE: Personal teachers list page

DESC: Given that a student is logged in, then the student should be able to view his/her personal teachers list page, sorted by department and course.

RAT: For a student to see his/her personal teachers list page.

DEP: FR4

**3.2.2 User Class 2 – The Teacher**

**3.2.1.1 Functional requirement 1.1**

**ID: FR1**

TITLE: Download the mobile application

DESC: A teacher should be able to download the mobile application through either an application store or similar service on the mobile phone. The application should be free of charge.

RAT: For a teacher to download the mobile application.

DEP: None

**3.2.1.2 Functional requirement 1.2**

**ID: FR2**

TITLE: Download and notify teachers of new releases

DESC: When a new/updated version or release of the software is released, the teacher should check for these manually. The download of the new release should be done through the mobile phone in the same way as downloading the mobile application.

RAT: For a teacher to download the most updated release.

DEP: FR1

**3.2.1.3 Functional requirement 1.3**

**ID: FR3**

TITLE: Teacher registration – Mobile and Web application

DESC: Given that a teacher has downloaded the mobile application or has browsed to the platform’s homepage, then the teacher should be able to register through the mobile application or the website. The teacher must provide first name, last name, password, e-mail address, university and the personal code he received from the university.

RAT: For a teacher to register on the mobile application or the website.

DEP: FR1

**3.2.1.4 Functional requirement 1.4**

**ID: FR4**

TITLE: Teacher log-in – Mobile and web application

DESC: Given that a teacher has registered, then the teacher should be able to log into the application. The teacher’s log-in information will be stored on the phone or browser, and in the future the teacher should be logged in automatically.

RAT: For a teacher to be logged-in after he/she is registered.

DEP: FR1, FR3

**3.2.1.5 Functional requirement 1.5**

**ID: FR5**

TITLE: Retrieve password

DESC: Given that a teacher has registered, then the teacher should be able to retrieve his/her password by e-mail.

RAT: For a teacher to retrieve his/her password.

DEP: FR4

**3.2.1.6 Functional requirement 1.6**

**ID: FR6**

TITLE: Mobile/Web application - Homepage

DESC: Given that a teacher is logged in to the application, then the first page that is shown should be the homepage, with the teacher’s profile picture on the navigation bar.

RAT: For a teacher to see his/her personal homepage.

DEP: FR4

**3.2.1.7 Functional requirement 1.7**

**ID: FR7**

TITLE: Personal profile page

DESC: Given that a teacher is logged in, then the teacher should be able to view his/her personal profile page.

RAT: For a teacher to see his/her personal profile page.

DEP: FR4

**3.2.1.8 Functional requirement 1.8**

**ID: FR8**

TITLE: Personal course page

DESC: Given that a teacher is logged in, then the teacher should be able to view his/her personal courses page, sorted by universities and departments.

RAT: For a teacher to see his/her personal course page.

DEP: FR4

**3.2.1.9 Functional requirement 1.9**

**ID: FR9**

TITLE: Free-text search

DESC: Given that a teacher is logged in, then the teacher should be able to conduct a search by providing either course name, first name, last name or university name. The results are displayed in a list view by default.

RAT: For a teacher to search through the free-text search.

DEP: FR4

**3.2.1.10 Functional requirement 1.10**

**ID: FR10**

TITLE: Sorting results

DESC: Given that a teacher has conducted a search, the teacher should be able to sort the results according to the result type: teacher, student, course or university.

RAT: For a teacher to sort his/her search.

DEP: FR9

**3.2.1.10 Functional requirement 1.10**

**ID: FR10**

TITLE: Filtering results

DESC: Given that a teacher has conducted a search, the teacher should be able to filter the results according to the result type: teacher, student, course or university.

RAT: For a teacher

to filter his/her search.

DEP: FR9

**3.2.1.11 Functional requirement 1.11**

**ID: FR11**

TITLE: Personal students list page

DESC: Given that a teacher is logged in, then the teacher should be able to view his/her personal students list page, sorted by course.

RAT: For a teacher to see his/her personal students list page.

DEP: FR4

**3.2.1.12 Functional requirement 1.12**

**ID: FR12**

TITLE: Message all students in course

DESC: Given that a teacher is logged in, then the teacher should be able to broadcast a message to all of his/her students in a selected course.

RAT: For a teacher to broadcast a message to his/her students in a certain course.

DEP: FR4