ISSN: 2454-1907
DOI: https://doi.org/10.29121/ijetmr.v5.i6.2018.245

IJETMR
International Journal of Engineering Technologies and Management Research
A Knowledge Repository

oPESA: ONLINE PLATFORM FOR AUTOMATIC EXAM-HALL SEAT ALLOCATION

Evi Papaioannou [*1], Angelos Vardakis [2], Christos Kaklamanis [1]
[*1] Department of Computer Engineering and Informatics, University of Patras & CTI "Diophantus", Greece
[2] Department of Computer Engineering and Informatics, University of Patras, Greece

Abstract:
We present oPESA, an online examination management system for exam-hall seating arrangement. oPESA is a platform built on PHP, HTML, CSS, JavaScript, jQuery and SQL. It is an open platform which can be accessed only by registered users, i.e., professors and students. Professors provide as input course details like title, examination date and room; additional input, like predetermined lists of students or list of constraints regarding the seating plan, can also be given. Students simply register for course examinations by providing their student ID as input. Whenever a student registers for an examination, oPESA immediately responds by automatically assigning an exam-hall seat to the student. Seat assignment is performed by an unbiased algorithm ensuring randomness and fairness. Results appear online but can also be downloaded and saved locally in .xls format. oPESA follows a responsive web design which allows it to automatically resize, hide, shrink or enlarge in order to look good on all devices (desktop computers, tablets, smartphones). oPESA suggests a very useful management tool for modern - highly-populated - higher education institutes which significantly facilitates the examination management process offering clarity and efficiency. Furthermore, oPESA could be also used for more general seat allocation purposes in the context of scientific, social and cultural events.

Keywords: Web-Based Platform; Efficient Management of Examinations; Automatic Seat Allocation; Responsive Application; Html5; Php; Sql.

Cite This Article: Evi Papaioannou, Angelos Vardakis, and Christos Kaklamanis. (2018). "OPESA: ONLINE PLATFORM FOR AUTOMATIC EXAM-HALL SEAT ALLOCATION." *International Journal of Engineering Technologies and Management Research*, 5(6), 51-65. DOI: https://doi.org/10.29121/ijetmr.v5.i6.2018.245.

1. Introduction

Management of course examinations in higher education institutes (i.e., university departments, colleges, etc) is an effort-consuming process. Given the constantly increasing number of students, the complexity of this process is also increasing in terms of both time and effort required for an efficient outcome. During examination periods, it is often the case that large groups of students massively enter exam halls struggling for a seat. A usual, yet not quite efficient approach, involves professors and/or supportive personnel indicating an "on-the-fly" seat assignment. However, such an approach usually brings about nervousness and intensity, which certainly disturb the

ISSN: 2454-1907
DOI: 10.5281/zenodo.1304431

smoothness of the examination procedure. An alternative approach is that students individually select their seat, in lack of a general plan. However, this could also result in an inappropriate examination process and, probably, in unfair examination outcomes, especially in cases where certain ethical conditions/rules are not fully respected.

Responding to such inefficiencies observed during examinations, several suggestions have been made recently regarding methods and applications for automatic exam-hall seat allocation. In [1], a C/C++-based seating tool is presented which automatically places students in their respective seats according to their allotted roll numbers, serially, in numeric order. Rooms are considered as a multi-dimensional array with specified number of rows and columns. Students are placed in each seat, one behind the other, according to last digits of their enrollment number. In [6], a database management system is suggested for the seating of students in examination halls. In [4], a PHP, web-based application for exam seating arrangement is described focusing on preventing cheating in exams. Rules for exam seating can be set based on number of students, capacity of seats, environment and exam type. In particular, depending on whether the room is rectangular or square, two types of seating methodologies are used for exam seating, resembling the letters X and I, respectively. In [3], an examination hall and seating arrangement application using PHP is suggested which also generates reports for facilitating administrative tasks by reducing paperwork. The described Dashboard software is based on the existence of an administrator who can create accounts, assign roles to users and grant privileges. The administrator can also create courses and generate exam hall and seat allocation plans which can be subsequently printed in the form of a report. ExamHall [5] is an android seating arrangement application built on PHP and JAVA. The application uses a database containing data like overall student register numbers, number of halls available, number of desks available in a particular hall and subject code. Files in this database are regularly updated by an administrator. Students are placed sequentially in a seating pattern based on their register number. In [2], the authors propose, analyze and evaluate a genetic algorithm for scheduling as a potential solution to a heavily constrained version of the seat allocation process. The objective of the work is to address the problem of allocating students to branches of higher education institutes as per their choice and rank scored in entrance examination. In [7], a web-based application is presented for the management of both the examination room assignments and the examination proctor assignments in each room.

Exploiting advances in internet and web-technology, we designed and implemented oPESA, an online examination management system for exam-hall seating arrangement. oPESA is essentially an HTML5 platform built on PHP, HTML, CSS, JavaScript, jQuery and SQL. It is an open platform which can be accessed only by registered users. There are two user roles: professors and students. Professors provide as input course details like title, examination date and room; additional input, like predetermined lists of students or list of constraints regarding the seating plan, can also be given. Students simply register for course examinations by providing their student ID as input. Whenever a student registers for an examination, oPESA immediately responds by automatically assigning an exam-hall seat to the student. Seat assignment is performed by an unbiased algorithm ensuring randomness and fairness. Results appear online but can also be saved locally in xls format. oPESA follows a responsive web design which allows it to automatically resize, hide, shrink or enlarge in order to look good on all devices (desktop computers, tablets, smartphones). oPESA suggests a very useful management tool for modern - highly-populated - higher education institutes which significantly facilitates the examination management process

ISSN: 2454-1907
DOI: 10.5281/zenodo.1304431

offering clarity and efficiency. Furthermore, oPESA could be also used for more general seat allocation purposes in the context of scientific, social and cultural events. The platform is currently available at http://bit.do/examz

The rest of the paper is structured as follows. In Section 2, we provide a detailed technical description of oPESA addressing design and implementation issues and approaches. In Section 3, we demonstrate oPESA functionalities via comprehensive screenshots. We discuss future plans in Section 4.

2. Materials and Methods

In this section, we provide technical details for oPESA. oPESA was developed using the LINUX operating system (version CentOS). Additional necessary software components include Web Server (e.g., Apache HTTP Server), Database (e.g., MySQL or MariaDB), PHP (version 7 or higher), Composer (latest version) and Php Spreadsheet, which is usually installed via the Composer.

Different technologies were used for the development of the platform which is essentially composed of 3 basic parts: Database, Server Side, Client Side. An orchestrated interaction of these parts is crucial for the correct function of oPESA as well as for a successful user experience. Below, we describe these parts in detail.

2.1. Database

The DBMS used for the oPESA is MySql. MariaDB makes a good alternative since the two systems are compatible. oPESA database contains 6 basic tables. Table “usr” contains users and information like username, encrypted password, account status (active/inactive), administration privilege (active/inactive), teacher property (active/inactive), the IP of the most recent user connection, user full name. Table “classroom” contains examination room ID as well as the username of the user who first stored it. Table “course” contains the courses. For each course, the following information is maintained: course name, course teacher, examination room, examination start and end time, course status (active/inactive) as well as a list of students who can participate in the examination. Table “student list” contains the list of students who can participate in the examination of particular courses. There is an M:N relation between student ID and course name. So, each record of the table “student list” has a composite primary key which is composed of student ID and course name (for all course examinations where a student can participate). Table “seat” contains all seats for all courses and includes information about the course examination for which a particular seat can be used, x and y coordinates, status (available/non-available/reserved/student ID). Table “aspect” is essentially the exam hall plan. This table provides the guide for the creation of elements in the table “seat”.

The entity-relationship diagram of our database is depicted in Fig. 1. A green background indicates a primary key while arrows indicate dependencies.

ISSN: 2454-1907
DOI: 10.5281/zenodo.1304431

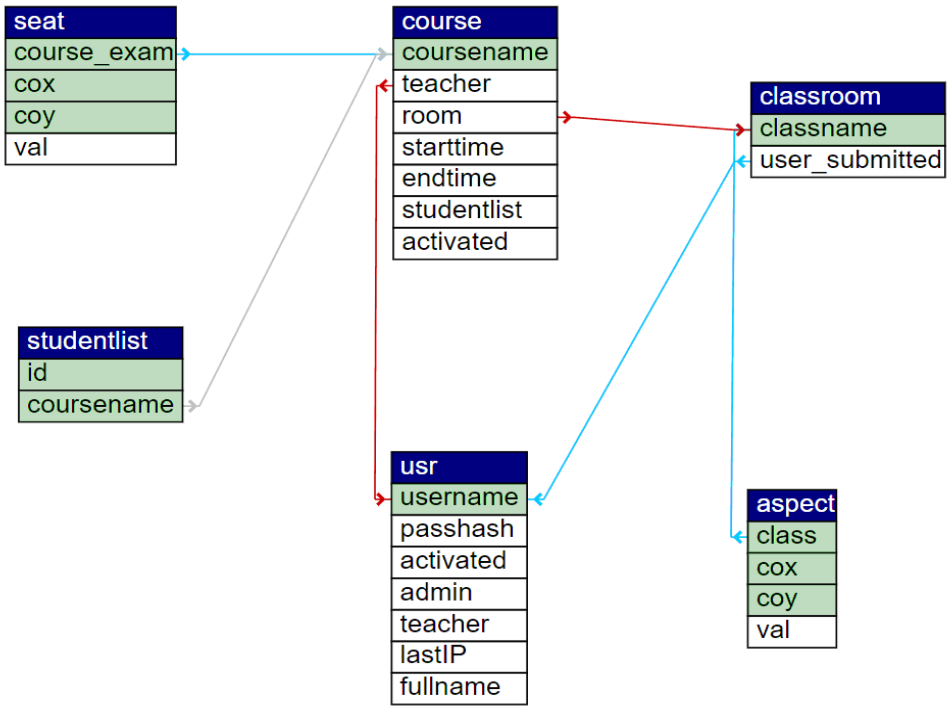

Figure 1: The entity-relationship diagram of oPESA database. A green background indicates a primary key while arrows indicate dependencies

Furthermore, the following conditions hold. Deletion of a teacher from the table "usr" implies the deletion of all information affiliated to this particular teacher like rooms, courses and related student lists, hall seats and plans. Deletion of a room implies the deletion of its plan, seats and affiliated courses. Finally, deletion of a course implies deletion of all "studentlist" entries containing the corresponding course name; it also implies that all seats available for this particular course are released.

2.2. Server Side

For the implementation of the server side part of oPESA, object-oriented PHP was used since it offers flexibility, code reusability and efficient grouping of functionalities and data.

The general communication model "Database ↔ Server Memory ↔ User Interface" has been adopted because it supports grouping of functionalities and data and also facilitates management and scalability of the platform. This scheme implies that data stored in the oPESA database may also reside to the server memory. New entries as well as updates or deletions of existing entries related to rooms, courses and users take place both in the server memory (e.g., during an active user session) and in the system database so that such data is immediately available to other users. oPESA is based on three prerequisites files: (a) core.php which contains the core code for oPESA including classes, their members and methods, (b) consts.php where all constants are defined thus making oPESA a highly parameterized system and (c) support.php which contains auxiliary functionalities frequently used within the system code. More precisely, core.php which is actually the cornerstone of the application has been built following a bottom-up approach. It contains 4

ISSN: 2454-1907
DOI: 10.5281/zenodo.1304431

major classes namely, "seat", "formation", "user" and "student". Objects of the class "seat" can represent either a position in an empty hall plan or a position in an actual hall plan for an examination. The class "seat" has 3 members, namely coordinates x and y as well as a value which indicates whether a particular seat is reserved (-1), non-available (0) or available (in this case the value can be assigned a student ID). The class "formation" is actually the basis where the examination of a course in a particular rooms is built. It contains information on course name, classroom ID, course and room seats, maximun number of seats horizontally and vertically for each room plan. It contains 2 major subclasses, namely "course" and "classroom". Objects of the class "user" represent a user who can currently be only a teacher/professor; however, the platform can be easily extended to accommodate also students as registered "users". It currently contains 1 subclass named "teacher". In addition to user information like username, password, full name, status and the most recent IP by which the user connected to the system, a timestamp of the last log activity of the user is maintained (*$lastSeen*) which is also used by session functions for security purposes. Objects of the class "user" represent students and contain information on student ID, list of eligible courses and timestamp for log activity.

In addition to these three basic components, there are also several PHP scripts which manage user requests and respond to them. PHP scripts should also be considered important components of oPESA since they play a significant role as intermediaries between the user interface and the server. PHP scripts of oPESA perform tasks like initialization of the system database (1streg.php), listing and deletion of rooms (delclassroom.php), login details (details.php), user management (edittch.php, keepalive.php, loginstd.php, logintch.php, logout.php, newusr.php), exam hall plan creation and export (exceltoroom.php, generateexcel.php), course management for teachers and students (getcourses.php, new_course.php), room management (newclass.php).

Important Server-Related General Issues

Data update and synchronization: The activity of the user (whether active or not) is recorded at regular intervals. In addition, the activity of teachers is updated after each request they make to the server. Active user logging is very important. It increases the security of the session as there is client-server communication and, therefore, agreement for the duration, renewal and termination of the session; furthermore, user data is continuously updated, since it is loaded directly the Database, thus providing up-to-date information on the data a user views and processes. Student data updates can take place less frequently (than after every request), so as to avoid mass queries to the database.

Random and unbiased seat assignment: the assignment of a hall seat to a student when enrolling in a course is based on the use of the rand () function. However, since rand () produces pseudo-random numbers, the srand function is actually used, as part of an algorithm which uses as argument a function including current time timestamps, start and end time of the test, MAX_SESS_DESTRUCT constant and the random_int function. In this way, a good level of randomness is guaranteed in the seat assignment process.

In view of the above, the prevention of possible malicious co-operation among students could fail either due to possible leakage of information or due to student coalitions (e.g., a student could maliciously connect with many IDs and thus gain extended knowledge on the exam seating plan).

ISSN: 2454-1907
DOI: 10.5281/zenodo.1304431

However, the risk induced by such issues can be completely diminished if empty rows and / or columns are used to provide an appropriately sparse seating plan.

Platform Security: it is extremely difficult to claim that a platform, application or website is generally sufficiently secure. Issues that need to be taken into account in order to fill all possible safety gaps are really numerous. However, the following tasks can significantly increase security. Server operating system and software certainly need to be updated as often as possible. Additionally, configuring the server to run applications on the HTTPS service with the necessary signed certificates certainly adds an additional level of protection.

Furthermore, it would be very useful to install a Web Application Firewall such as ModSecurity, which via appropriate rules can protect the system from malicious requests and attacks of various types.

User passwords are stored in the Database using the bcrypt encryption algorithm. The security it provides is satisfactory given the purpose of the platform. However, it is relatively easy to use the Argon2 encryption algorithm (for PHP versions 7.2 or higher).

At the session level, appropriate functions have been defined to reduce the risk of session fixation / hijacking. More specifically, only one IP per session is allowed. So, in the case of a request made by a client connected to an already open session, the system deletes the session and unlinks the client. Also, the id of a session is regenerated regularly so that session ID interception and reuse is avoided.

2.3. Client Side

The user interface forms the last part of the platform communication chain containing all data that is to be processed, displayed and stored. The server files accessible via hyperlinks include admin_panel.php, courses.php, first_reg.php, home.php, index.php, newcourse.php. At the beginning of most of these files there is a PHP script that checks whether the user is logged in and is a teacher or student before deciding whether or not to display the page. Also, all files include support.js that contains auxiliary functions. Additionally, admin_panel.php includes adminpanel.js where admin-specific utilities are implemented.

2.3.1. Layout Design

The site has been built on Bootstrap 4, providing a responsive design and user-friendly graphics environment. At the same time, several components of Bootstrap have been used, like for example alerts, buttons, dropdown menus, forms, input groups, list groups, modals, navbar, progress, tooltips.

Of course, HTML and CSS have been also extensively used. Custom items and parts were also created where appropriate. For instance, a CSS specifically formatted for cells was developed for the creation of seats. In this way, space on the user's screen is saved since table elements occupy the minimum possible space in relation to other elements.

ISSN: 2454-1907
DOI: 10.5281/zenodo.1304431

2.3.2. Interaction Via the User Interface

Users can modify the position of seats in an exam hall so that they meet their requirements. Furthermore, administrators can create exam halls from scratch and modify all of their parameters. Such functionalities require scripts to dynamically modify and add DOM document elements. For this reason, jQuery and JavaScript have been used in all pages accessed by users.

jQuery - via the use of few lines of code - facilitates the definition of actions that must be performed if an event takes place. In general, it is a powerful tool for batch selection and modification of several elements for which sophisticated conditions have been set.

Additionally, 2 plugins have been added to facilitate data entry while providing a user-friendly visual result: (i) jQuery Date and Time picker which allows users to select date and time to be automatically entered in the input field, (ii) Bootstrap Toggle, which is actually a user-friendly checkbox with the graphics of an on-off switch.

2.3.3. Sending Requests and Communicating with the Server

The interactivity of the platform is largely due to asynchronous communication via AJAX. In this way, users are not required to switch from page to page to complete an action or operation. JSON has been selected to perform data send and receive due to simple syntax and structure.

Each client request is sent either in JSON format or through the API FormData (when files must be uploaded together with data). In the latter case, feedback is provided on the progress of the file upload through the progress bar.

The request method is always POST when referring to sending data to the server since it is safer than e.g., GET which sends data through the request URL. However, GET is used when just a request without data must be sent. For example, to regularly send an active session request to the user via the keepalive function specified in support.js, the method GET is used.

In general, all requests (other than those mentioned above) to the server are accompanied by a server response, augmented - when needed – by a success/failure message.

3. Results and Discussions

oPESA is currently available at http://bit.do/examz. Below, we provide a detailed description of several of the functionalities offered by oPESA.

When visiting oPESA, users are prompt to log in to the system either as Teachers or as Students (Fig. 2).

ISSN: 2454-1907
DOI: 10.5281/zenodo.1304431

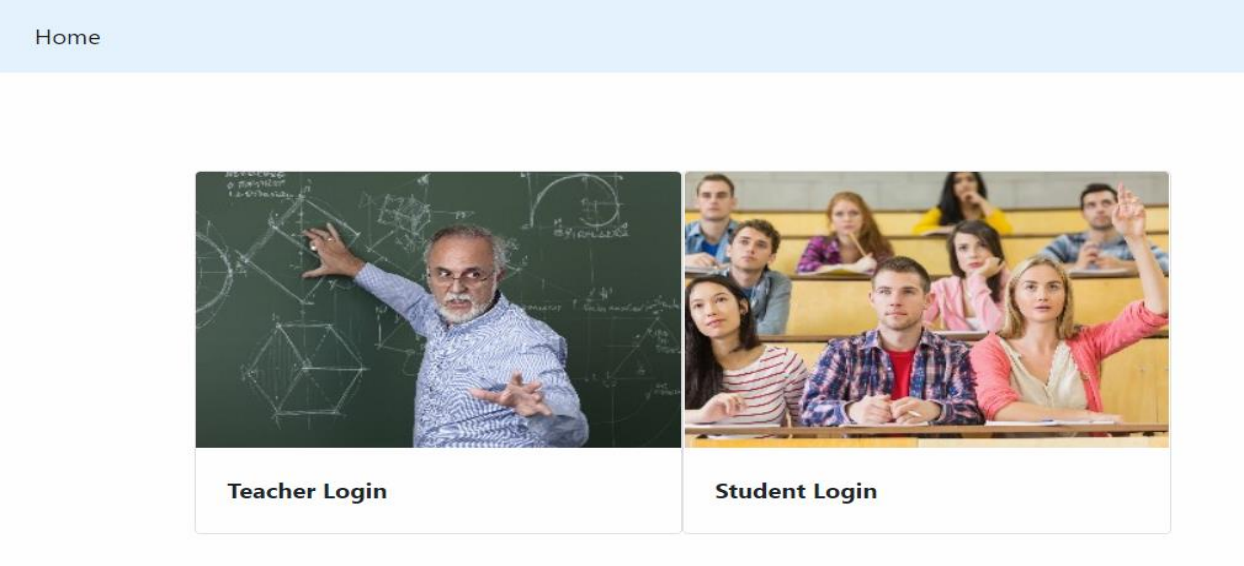

Figure 2: oPESA log-in screen

A user wishing to connect as a teacher is prompted to provide a Username and a Password (Fig. 3(a)). In case wrong credentials are provided, an appropriate message is returned (Fig. 3(b).

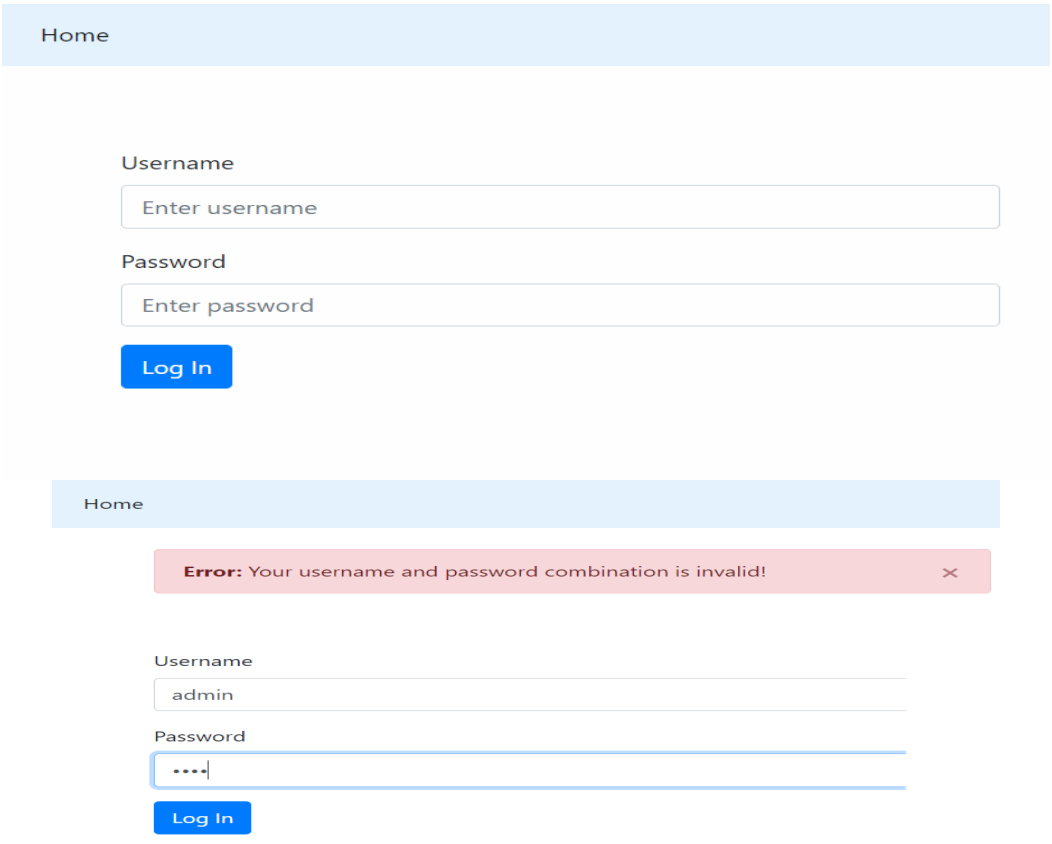

Figure 3: (a) Teacher log-in screen, (b) Error message caused by wrong credentials

ISSN: 2454-1907
DOI: 10.5281/zenodo.1304431

Successful login is confirmed via a message appearing for 1.5 sec. Then, the user is automatically redirected to the oPESA home page with links to "Saved Courses, "New Course", "Admin Panel" and a "Log Out" option.

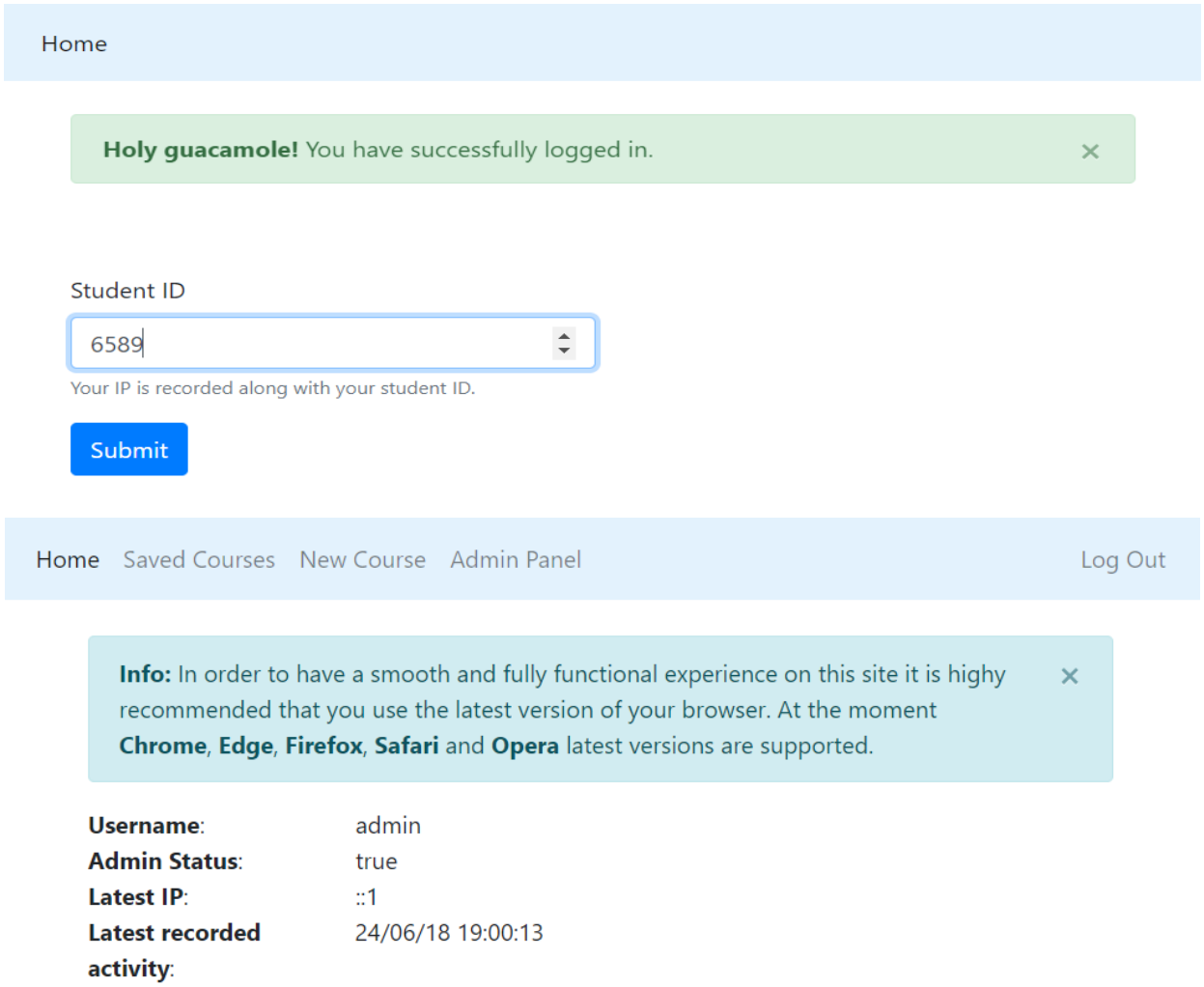

Figure 4: (a) Successful log-in, (b) Home page for a teacher

A teacher can organize the examination of a new course selecting the option "New Course" from the horizontal menu. For the creation of a "New Course" the following information is required (Fig. 5(a)): course name, classroom (from an existing list, Fig. 5(b)), exam start date and time, exam duration and status (open/closed). Optionally, a teacher can upload a .xls file containing IDs for students who have the right to participate in the examination of this particular course; if this is done, then only students whose ID is included in this file can register for the specific examination. Otherwise, all students can register for the exam. Furthermore, a teacher can modify the status of seats in an examination hall. Seat status is visually indicated: green cells indicate available seats; gray cells indicate non-available seats (Fig. 6). The status of all seats are reset (all available/non-available) by clicking on the # at the beginning of the table. Exam start date and time can be chosen on a dropdown calendar.

ISSN: 2454-1907
DOI: 10.5281/zenodo.1304431

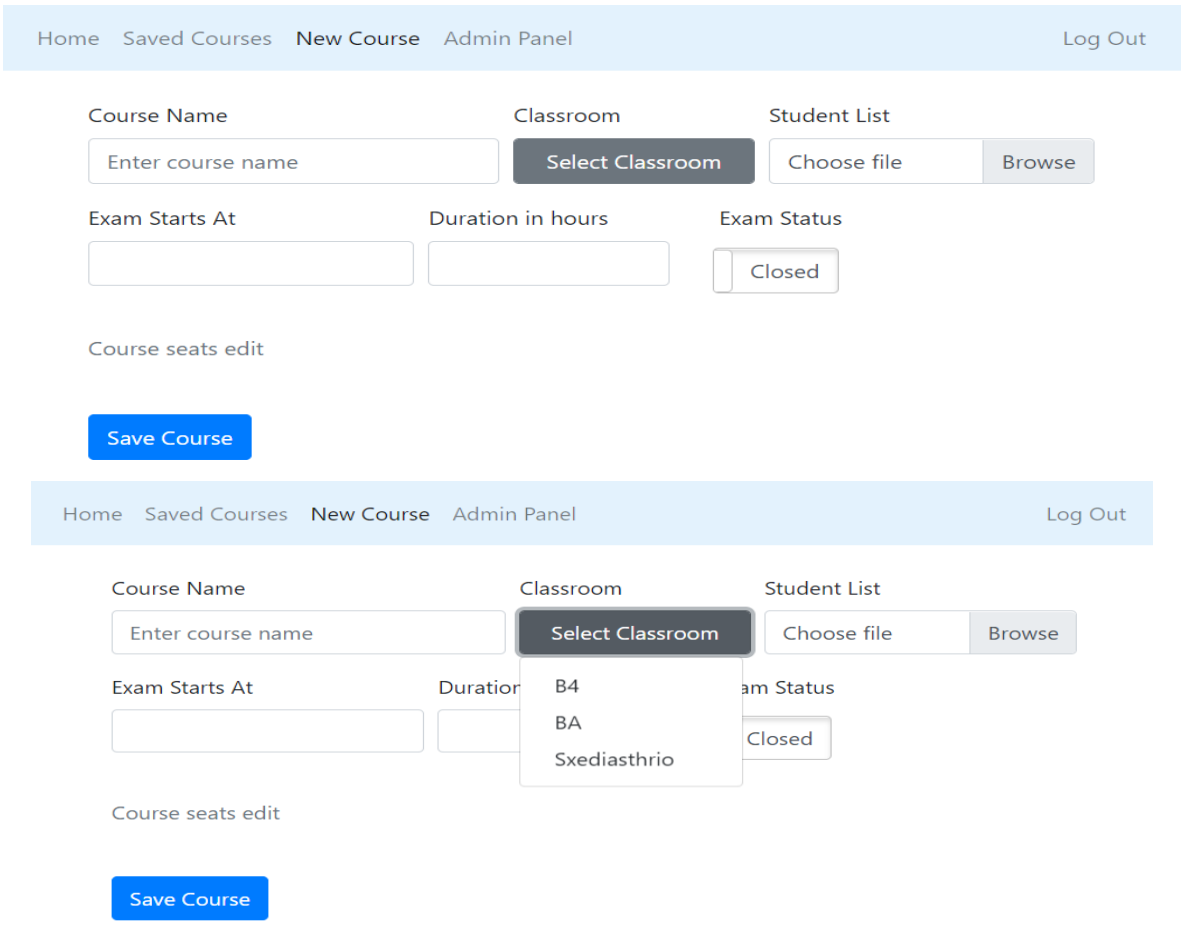

Figure 5: (a) Creating a New Course, (b) Selecting the exam hall for a New Course

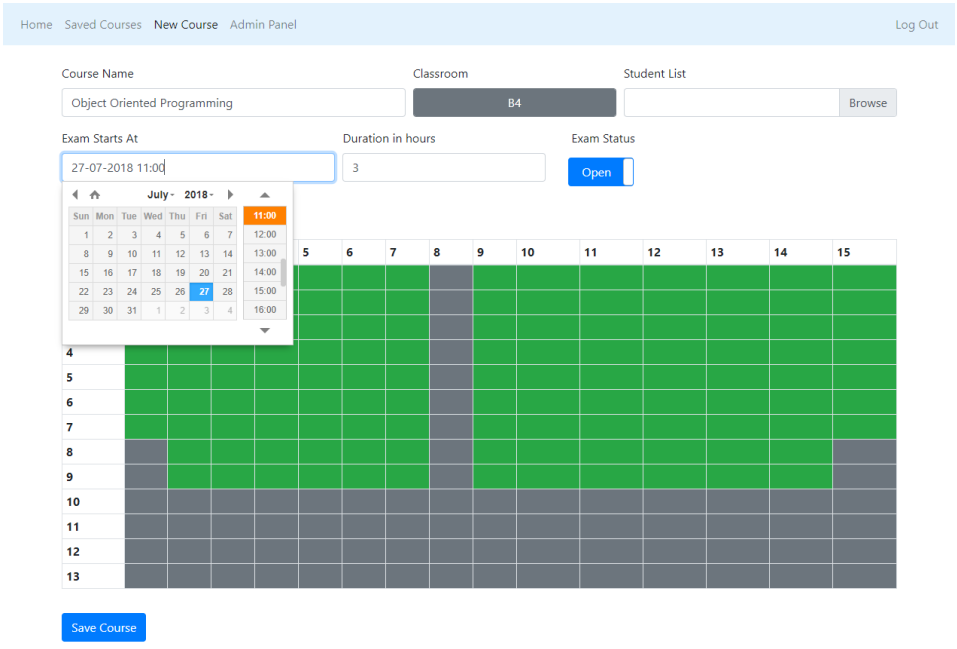

Figure 6: Fixing the details for the examination of a "New Course"

ISSN: 2454-1907
DOI: 10.5281/zenodo.1304431

Once all details are set on the corresponding form, clicking on the blue button "Save Course" (Fig. 7) completes the process.

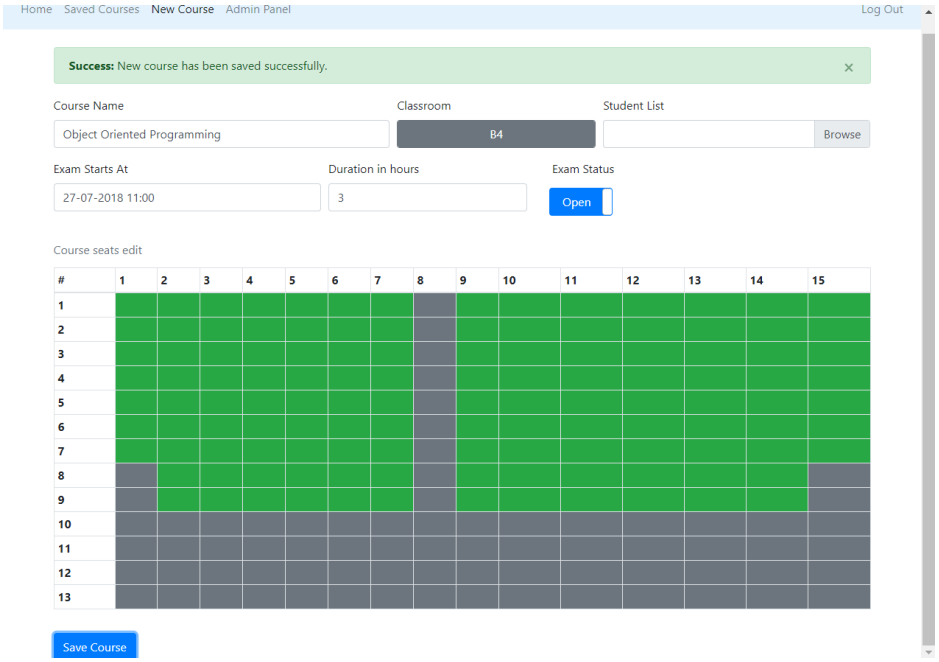

Figure 7: Successful creation of a "New Course"

Via the option "Saved Courses" in the horizontal menu, a teacher can (i) retrieve or modify information about all scheduled courses or delete courses (Fig. 8), (ii) export course examination details including exam hall seat allocation in an .xls file (Fig. 9).

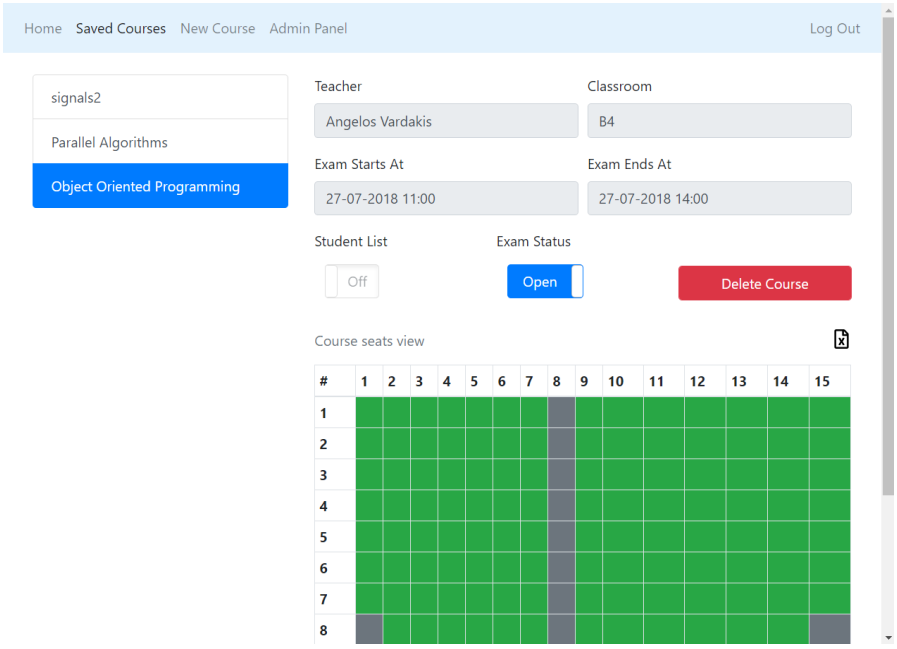

Figure 8: Management of course examination details

ISSN: 2454-1907
DOI: 10.5281/zenodo.1304431

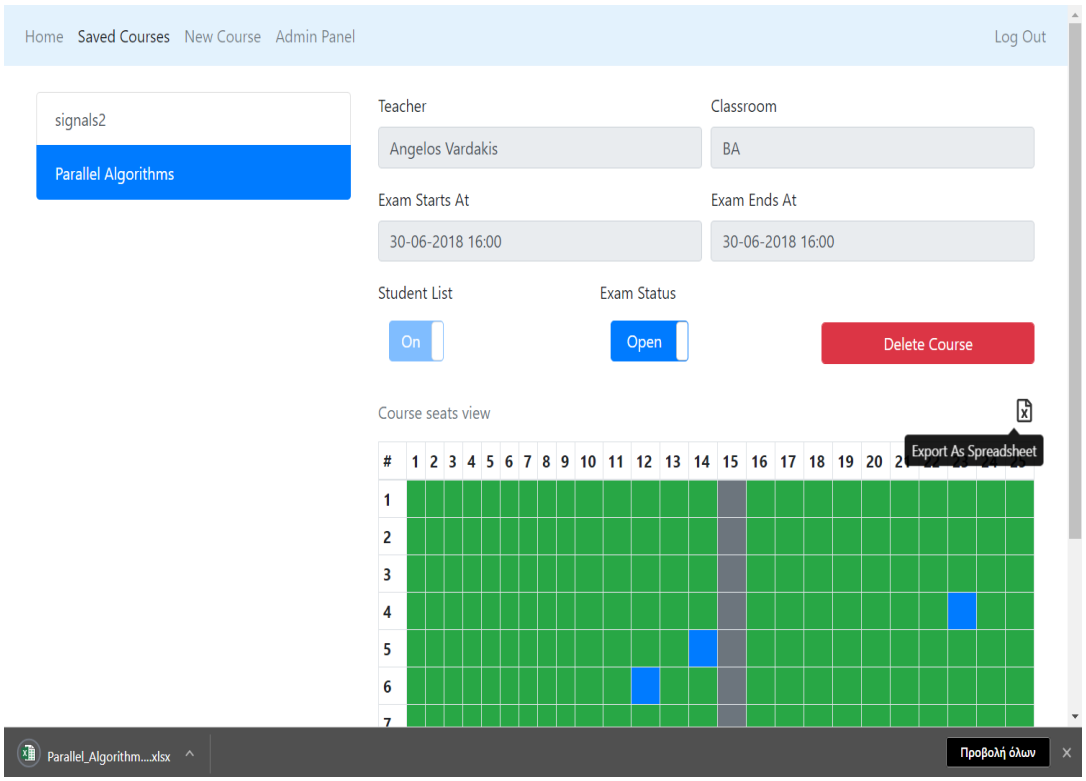

Figure 9: Export course examination details

The option "Admin Panel" allows a user with administrative privileges (currently a teacher) to perform administrative tasks like adding a new teacher or modifying details of an existing one, as well as adding a new exam hall or removing an existing one (Fig. 10).

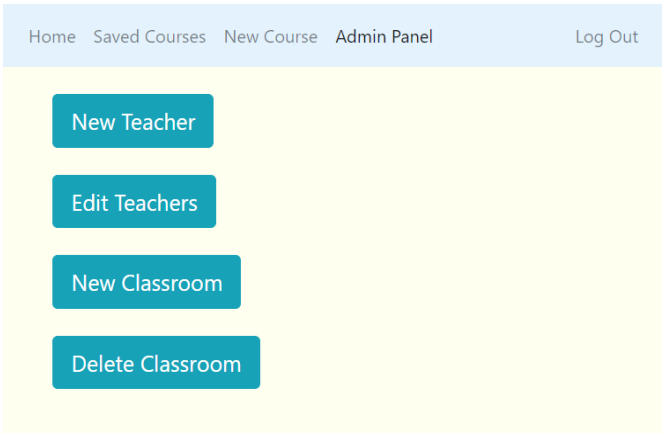

Figure 10: Available administrative tasks

For the addition of a new exam hall, parameters like name, dimensions and availability of seats can be set. Furthemore, the hall plan can be directly exported as a .xls file (Fig. 11).

ISSN: 2454-1907
DOI: 10.5281/zenodo.1304431

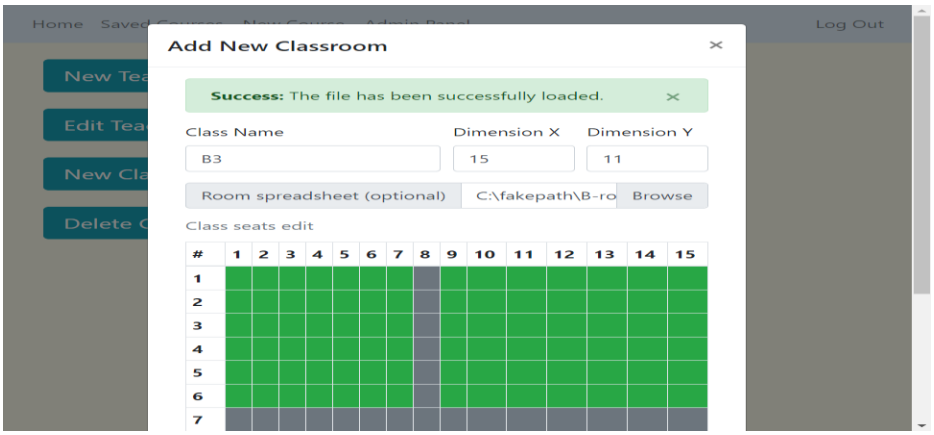

Figure 11: Adding a new exam hall

A user wishing to connect to oPESA as a student is prompted to provide a Student ID (Fig. 12). Then, a list of courses appears where the Student has the right to register for exams (Fig. 13).

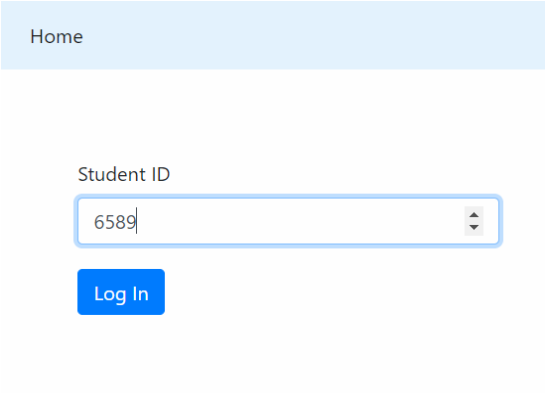

Figure 12: Student login

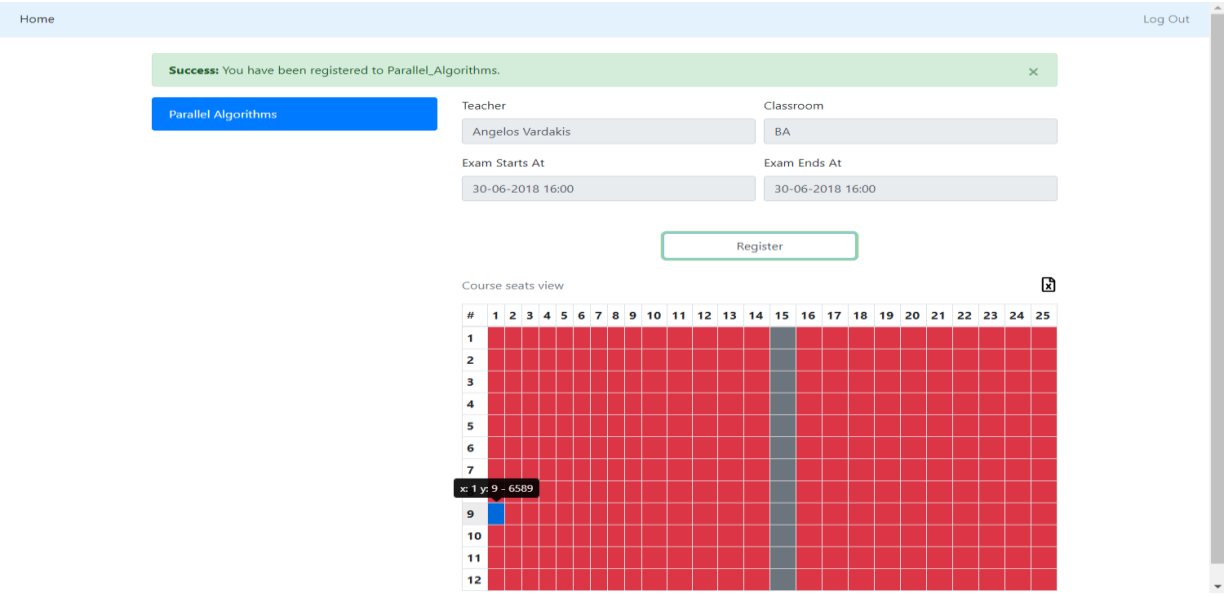

Figure 13: Student registration for exams

ISSN: 2454-1907
DOI: 10.5281/zenodo.1304431

For each available course, Students can retrieve details like course name, professor, examination room, exam date, time and duration. Students can register for participating in the course examination and be immediately informed about the exam-hall seat assigned to them, together with the exact seat coordinates (Fig. 13). All information can be exported to an .xls file (Fig. 14).

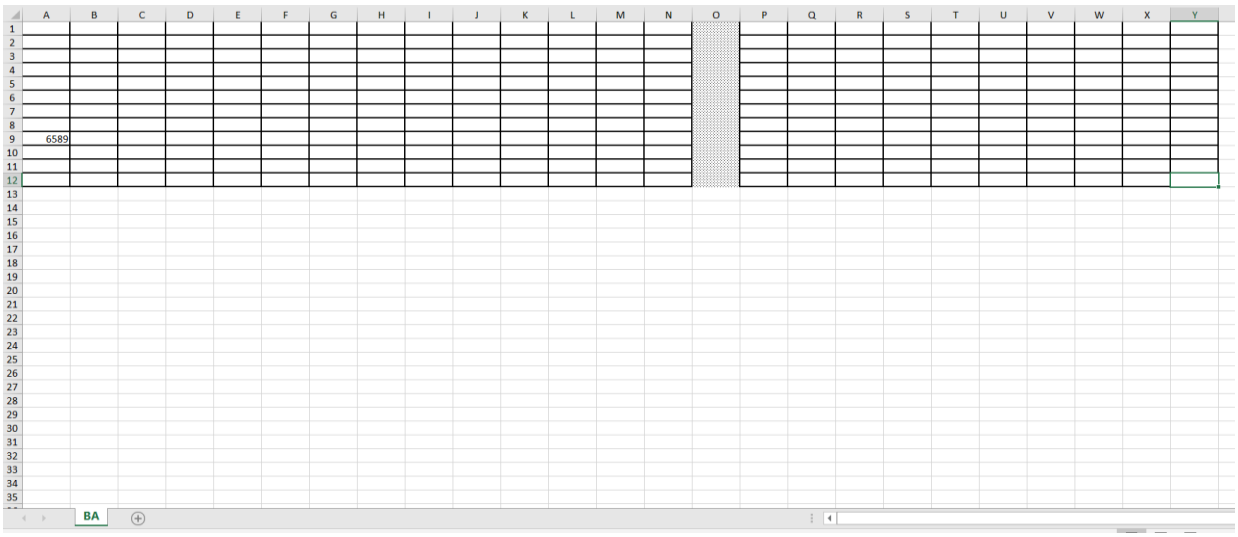

Figure 14: .xls file for student seat in an examination hall

4. Conclusions and Recommendations

Exploiting advances in internet and web-technology, we designed and implemented oPESA, an online examination management system for exam-hall seating arrangement. Although initially intended for facilitating the management of course examinations in higher education institutes, oPESA can be also used for more general seat allocation purposes in the context of scientific, social and cultural events.

oPESA is an open platform, publicly available at http://bit.do/examz which can be accessed only by registered users. It is built on PHP, HTML, CSS, JavaScript, jQuery and SQL and follows a client-server approach. Special attention has been given to security issues as well as to issues related to the unbiased, truly random seat allocation. oPESA features a responsive design which makes it an attractive, easy-to-use application to be exploited not only by desktop users but also by users of modern, wireless devices like smartphones or tablets. Furthermore, .xls files can be used for providing input or storing oPESA results.

Currently, oPESA is openly available but has only been tested in vitro. It is within our plans to use the oPESA platform in practice during the next examination period in at least two departments of the University of Patras and obtain feedback for evaluation. We also hope that external users also exploit oPESA and provide feedback for its functionality and user-friendliness. Towards this aim, we plan to design and launch an open evaluation process initially addressed to professors and students in higher education institutes.

ISSN: 2454-1907
DOI: 10.5281/zenodo.1304431

References

[1] Alam, A. F. Automatic seating arrangement tool for examinations in universities/colleges. International Journal of Engineering Applied Sciences and Technology (IJEAST), Vol. 1, Issue 4, ISSN No. 2455-2143, 2016, 8-10.
[2] Chandel, A., Sood, M.. A Genetic Approach Based Solution for Seat Allocation during Counseling for Engineering Courses. International Journal on Information Engineering and Electronic Business, vol. 1, 2016, 29-36.
[3] Gokila, R., Rohan Dass, A. Examination Hall and Seating Arrangement Application using PHP. International Journal of Engineering Science and Computing (IJESC), Vol. 8, Issue 2, 2018, 16059-16065.
[4] Nikam, N., Jagdale, A., Patil, G., Patil, P. Algorithm for efficient seat allocation process in college exam system. International Research Journal of Engineering and Technology (IRJET), Vol. 04, Issue 03, 2017, 2844-2851.
[5] Sowmiya, S., Sivakumar, V., Kalaimathi, M., Kavitha, S. V. Automation of exam hall seating arrangement. International Journal of Advance Research, Ideas and Innovations in Technology (IJARIIT), Vol. 4, Issue 2, 2018, 1048-1052.
[6] Thampan, M. Examination Hall Allocation. Engineering, SlideShare, Aug 10, 2014 (https://www.slideshare.net/martinathampan/examhall-allocation).
[7] Vasupongayya, S., Noodam, W. Kongyong, P.. Developing Examination Management System: Senior Capstone Project, a Case Study. International Journal of Computer and Information Engineering, Vol. 7, No 7, 2013, 1046-1052.

*Corresponding author.
E-mail address: papaioan@ ceid.upatras.gr