

TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Cyber Security Engineering

Almaz Kydyrmin 194435IVSB

Building a Secure Web Application for the Self-Development Camps Program in Kazakhstan

Bachelor's thesis

Supervisor: Kaido Kikkas, PhD

Co-supervisor: Nursultan Akhmetov, MSc

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Küberturbe tehnoloogiad

Almaz Kydyrmin 194435IVSB

Turvalise veebirakenduse arendamine Kasahstani enesearenduslaagrite programmile

Bakalaureusetöö

Juhendaja: Kaido Kikkas, PhD

Kaasjuhendaja: Nursultan Akhmetov, MSc

Tallinn 2022

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Almaz Kydyrmin

10/04/2022

Abstract

Today automation is present in modern businesses of different sizes – including common software applications, and more obvious implementations like autonomous robots or self-driving vehicles. Automation helps people to lower operation costs, increase productivity, reduce outsourcing needs, and become more competitive in their field.

The goal of this bachelor’s thesis is to build and deploy software for the “Self-Development Camps” Program that would help organizers and participants to automate their flow. The solution needs to be deployed under the program domain and be accessible online. The application must implement OWASP Top 10 security requirements.

The web application was developed in a modern Single Page Application approach using PostgreSQL, Django and React. It allows teachers to check exams automatically by providing links to the answer key and exam results and seeing the most common mistakes done by students for each particular examination. For students, there is an opportunity to review all the previously taken exams. For an unauthorized user, there are options to read general information about the program, access the program management contacts, browse upcoming events, notify program managers about website issues or ask questions from organizers of the program.

To conduct a security audit OWASP Top 10 security standard was used and the outcome has been analyzed.

As a final result of this bachelor thesis work, a working single page web application compliant with OWASP Top 10 has been built and deployed to the Digital Ocean cloud service provider.

The thesis is written in English and contains 54 pages of text, 6 chapters, 12 figures and 3 tables.

Annotatsioon

Tänapäeval on automatiseerimine olemas erineva suurusega kaasaegsetes ettevõtetes, sealhulgas levinud tarkvararakendused ja ilmsemad rakendused, nagu autonoomsed robotid või sõidukid. Automatiseerimine aitab inimestel alandada tegevuskulusid, tõsta tootlikkust, vähendada allhangete vajadusi ja muutuda oma valdkonnas konkurentsivõimelisemaks.

Käesoleva bakalaureusetöö eesmärk on luua ja juurutada tarkvara enesearenduslaagrite jaoks. “Enesearenduslaager” on programm, mis aitaks korraldajatel ja osalejatel oma voogu automatiseerida. Lahendus peab olema juurutatud programmi domeeni all ja olema võrgus juurdepääsetav. Rakendus peab kasutama OWASP top 10 turvanõudeid.

Veebirakendus on tehtud kaasaegses ühelehelise rakenduse lähenemisviisis, kasutades PostgreSQL, Django ja React. See võimaldab õpetajatel eksameid automaatselt kontrollida, pakkudes lingid vastusevõtmele ja eksamitulemustele ning näha õpilaste enamlevinud vigu iga konkreetse eksami kohta. Õpilastel on võimalus üle vaadata kõik varasemad sooritanud eksamid. Volitamata kasutaja jaoks on võimalik lugeda üldist teavet selle programmi kohta, pääseda juurde programmihalduse kontaktidele, sirvida eelseisvaid sündmusi, teavitada programmi juhte veebisaidi probleemide kohta või küsida küsimusi programmi korraldajalt.

Turvaauditi läbiviimiseks kasutati OWASP turvastandardit ja selle tulemusi on analüüsitud.

Käesoleva bakalaureusetöö lõpptulemusena luuakse OWASP-iga ühilduv üheleheline veebirakendus ja võetakse see kasutusele Digital Oceani pilveteenuses.

Lõputöö on kirjutatud inglise keeles ja sisaldab 54 lehekülge teksti, 6 peatükki, 12 joonist ja 3 tabelit.

List of abbreviations and terms

API	Application Programming Interface
BIL	Bilim Innovation Lyceum
CI	Continuous Integration
CLI	Command Line Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
DRF	Django REST Framework
EU	European Union
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTPS	Hypertext Transfer Protocol Secure
IELTS	International English Language Testing System
IT	Information Technology
JS	JavaScript
JSON	JavaScript Object Notation
KarBIL	Karagandy Bilim Innovation Lyceum
MVC	Model-View-Controller
NIST	National Institute of Standards and Technology

ORM	Object Relational Mapping
OWASP	Open Web Application Security Project
PBKDF2	Password-Based Key Derivation Function 2
PCI DSS	Payment Card Industry Data Security Standard
REST	Representational State Transfer
SMTP	Simple Mail Transfer Protocol
SPA	Single Page Application
SSR	Server Side Rendering
SSRF	Server-Side Request Forgery
SQL	Structured Query Language
UI	User interface
VPN	Virtual Private Network
WSTG	Web Security Testing Guide

Table of contents

1 Introduction	10
1.1 Problem and background	10
1.2 Functional requirements	11
1.3 Methodology	14
1.3.1 Model-View-Controller	14
1.3.2 OWASP	14
1.4 Thesis overview	15
2 Technologies	16
2.1 Web development approaches	16
2.2 Backend	18
2.3 Frontend	20
2.4 Other technologies	22
2.4.1 Docker and docker-compose	23
2.4.2 Celery	23
2.4.3 Redis	24
2.4.4 PostgreSQL	24
2.4.5 Nginx	25
3 Implementation	26
3.1 IT Architecture	26
3.2 User flow	27
3.3 Database Architecture and Security	27
3.4 Server-Side App	28
3.5 Client-Side App	29
4 Security Audit	30
4.1 Overview of OWASP Top Ten	30
4.2 Validating security requirements	30
4.2.1 Broken Access Control	30
4.2.2 Cryptographic Failures	31
4.2.3 Injection	32
4.2.4 Insecure Design	33
4.2.5 Security Misconfiguration	34
4.2.6 Vulnerable and Outdated Components	34
4.2.7 Identification and Authentication Failures	35
4.2.8 Software and Data Integrity Failures	36
4.2.9 Security Logging and Monitoring Failures	38
4.2.10 Server-Side Request Forgery	39
4.2 Security audit results	39

5 Further Development	41
5.1 Schedule generator and event management	41
5.2 Teacher cabinet	41
5.3 Parent role	41
5.4 Automatic recommendations for the program students	41
5.5 Other subjects	42
5.6 Monitoring System	42
6 Conclusion	43
References	44
Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis	48
Appendix 2 – Screenshots of the application	49

1 Introduction

In this chapter the author describes the problem in detail and lists all the functional requirements, reports methodology and makes an overall overview on this bachelor thesis.

1.1 Problem and background

The “Self-Development Camps” Program is a 3-year-old project in Karagandy “Bilim-Innovation” lyceum (secondary school). The main goal of this program is to help students to improve their math and English language level in order to be able to study at the world’s top universities. The co-supervisor of this work is one of the organizers of the program. At this specific point in time, the program is popular among the students of KarBIL and has 200 participants. There are many schools willing to use the “Self-Development Camps” Program but the program needs to be automated first.

One of the requirements made by organizers was to automatically check the exams taken by the program participants by providing the correct answers and exam results URLs. Students need to be able to see all the previously taken exams and make a review of them by looking at questions, chosen answers and the correct answers. Everyone with internet access must be able to read general information about the program, access the program management contacts, browse upcoming events, notify program managers about website issues or ask questions from the program organizers.

Previously, automation in some parts of the project was done by a number of different applications including Google Forms, Google Sheets, Google Drive, Google Presentations, Google Docs, Telegram, and Autocrat Extension, a multi-purpose document merge tool that allows to take data from a spreadsheet and merge it into a document via a template [54]. The solution attempts to automate the remaining parts and unify the information flow. There is no suitable software that would handle all the needs of the project. Different applications, some of which require a subscription, can solve part of the problems but the school needs a unified

solution. In addition to that, using some of the Google products and the Autocrat Extension requires technical background that not every teacher might have.

Since the security side of the project was one of the project's emphasized requirements, the OWASP Top 10 security framework was chosen due to its popularity among the community and solid reputation.

The author of this thesis is the sole software developer involved in the project and was assigned the roles of security specialist and full-stack engineer.

1.2 Functional requirements

One of the functionalities of the project is to automate checking the academic English language exams of different levels (YLE, KET, PET, FCE, IELTS). For that teachers need to provide links to the Google Sheets with answer-key and answers of all the exam takers. Afterwards, teachers shall be able to see the general information about the exam, the most common mistakes, and a list of participants with scores in each exam section.

Students of the program must be able to review all the previous exam attempts and see questions, chosen answers, and correct answers for every question of the exam.

In addition to that, a homepage for the “Self-Development Camps” Program needs to be created to provide general information about the project including program description, location, organizers, subjects, price, and contacts of the organizers. Moreover, a feature to ask questions from organizers is needed.

The goal of this bachelor thesis work is to build and deploy a secure web application that satisfies the requirements of the OWASP security standards and meets the features defined in detail in the next sections.

1.2.1 Exam checker

Teachers must be able to create exams by providing links to the answer key and the results for the reading and listening sections (see Figure 8). The sheets need to be parsed and all the participants with an account in the application shall see the exam that they took with all the detailed information on their user profile page. Automatic checking shall happen in the background so that the web application is still available for other concurrent users.

1.2.2 Teachers' exams view

Teachers must be able to browse through the list of all the exams (see Figure 5). They must see the following data in the list view:

- exam identification number, date, and type;
- the number of participants.

By clicking on a specific exam teachers must be able to see the detailed information about it (see Figure 6), including

- the list of all the participants and their results by each section and in total;
- the list of the most common mistakes made for each of the exam sections and the number of how many times the question was answered wrongly;
- the number of participants;
- exam identification number and date;
- the maximum score in the exam.

1.2.3 Student profile view

Students must be able to browse through the list of all the exams they have participated in (see Figure 12). They must see the following data in the list view:

- exam identification number, date, and type;
- total score;
- the number of questions.

1.2.4 Exam attempt detail view

This view is shared by students and teachers. Teachers from the exam detail view can click on a specific student from the list of all the participants and see the details of the attempt. Students can click on a specific exam attempt available in their student profile and see the same data.

They must see the following data in the detailed exam attempt view (see Figure 7):

- question content, question number, chosen answer, correct answer and sign determining if the question was answered correctly for every question in each section of the exam;
- the name of the exam taker;
- exam date and type.

1.2.5 Homepage

Every user must be able to see the following general information about the program (see Figure 11):

- program description;
- location of the program;
- organizers of the program;
- subjects included in the program;
- price and start date of the program.

1.2.6 Contacts

User must be able to browse through the organizer contacts on the home page (see Figure 10).

Users must be able to see the following data:

- mobile phone number;
- email address;
- social media links if available.

1.2.7 Ask organizers

Users must be able to notify program managers about some website issues or ask a question anonymously from the home page of the web application (see Figure 10). Organizers must be able to see all the feedback from the admin panel including the content and the date of the feedback.

1.3 Methodology

To deliver secure software that meets all the requirements, the author had to analyze the functionality and choose the modern and appropriate technologies considering that there is a need to run tasks in a background mode and this application might be scaled to support other schools except KarBIL as well in the near future. The API was required to be modular and scalable, to make it easy to add more applications for the user interface as mobile apps for instance. To meet the security requirements, the author had to make a security check for compliance with the OWASP Top 10 security standard.

1.3.1 Model-View-Controller

The automation problem will be solved by building a web application: it is a modern way to provide quick access to information using different devices including cellphones, PCs, laptops, and tablets.

The product will be implemented according to the MVC Architecture because user interaction with an MVC application follows a natural cycle: the user takes an action, and in response, the application changes its data model and delivers an updated view to the user. And then the cycle repeats. This is a convenient fit for web applications delivered as a series of HTTP requests and responses [16].

1.3.2 OWASP

In today's world, there are automated tools that can test the security of a system or application. Automation often fails at a logical level when an application needs to be tested for business-logic flaws [17]. There are different frameworks that help organizations to make sure their software is secure and can be trusted by clients. OWASP is an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. All of the community projects, tools, documents, forums and chapters are free and open to anyone interested in improving application security [18].

The author has conducted a security audit according for compliance with OWASP Top 10 to ensure that the end product is as secure as possible.

1.4 Thesis overview

The first chapter of this work describes the details of all the functionality requirements and the problem background.

The second chapter of the thesis introduces different modern web technological stacks and explains the final stack choice decision.

The third part of the work describes the architecture and the engineering phase.

The fourth part of the thesis focuses on the project's security aspects, assesses the results and proposes future improvements.

The fifth chapter of the work describes plans for the future development of the project.

The sixth part of the thesis concludes the outcome of the work.

2 Technologies

There are numerous solutions to one specific problem in the world of information technologies. Each of them has its own benefits and drawbacks. In this chapter, the author compares different approaches and technologies and explains the reason for the final technological stack choice.

2.1 Web development approaches

There are two general approaches to building web applications today: traditional server-side rendering (SSRs) applications and single-page applications (SPAs). SSRs perform most of the application logic on the server: the rendering occurs on the server-side before passing them on to the browser. SPAs, however, perform most of the user interface logic in a web browser, communicating with the backend primarily using web APIs. Both approaches have their own advantages and drawbacks, which the author has presented in a detailed way in Table 1 [1], [2], [3].

Table 1. Comparison of web development approaches

	Server Side Rendering	Single Page Application
Usage	The application has simple, possibly read-only, client-side requirements. The application needs to function in browsers without JavaScript support. The team is unfamiliar with JavaScript or TypeScript development techniques.	The application must expose a rich user interface with many features. The team is competent in writing modern JavaScript using a SPA framework.
User Interface	Every page needs to be rendered on the server-side and the user needs to wait for that.	Ability to redraw portions of the screen dynamically like a desktop application, and the user sees the

		update instantly.
Transaction Payloads	Have the overhead of having to respond with the next page's content. Because the entire page is re-rendered, the content returned in traditional applications also includes HTML markup.	Transactions with the server are lighter and faster because, after initial delivery, only data is sent and received from the server.
Performance	Initial load may be faster, but if a new page is requested, the user needs to wait until the server renders a new page and it arrives at the user's browser.	Initial load may be longer, once the application has fully loaded, there will not be additional loading required. As the previous points state, screens render quickly and smoothly, and transactions are lightweight and fast. These characteristics all lead to less user wait time.
Development time	Only one application is needed to render the pages on the server's side.	Two separate applications need to be created. However, since the frontend communicated to the backend using APIs, the endpoints can be reused for other applications such as mobile or desktop.
Initial release	The MVC software architectural pattern has existed in one form or another since the 1970s, but it became more popular and generally accepted with its use in	The stage was set in the early 2000s. A brand-new way of thinking about web page design came about when the AJAX movement started to gain steam.

	web application frameworks such as Ruby on Rails, CakePHP, and Django.	
--	--	--

The following high-level conclusions about SPAs and SSRs can be drawn from the comparison shown in Table 1:

- Single page applications are used when a rich user interface with many features is required. This approach excels in performance, user interface, and user experience. However, users need to have JavaScript support in their browsers. In addition to that, this approach is widely used when the application must already expose an API for other (internal or public) clients.
- Server-side rendering is an old approach and is widely used when applications are primarily consumed in a read-only fashion by the vast majority of their users. It might be freezing for end users since they need to wait until the server renders the entire page and sends the markup to the browser.

SPA approach was chosen for the application due to its UI and performance advantages. In addition to that, there is a possibility that in future a mobile application will be released for the “Self-Development Camps” program and it could share the same APIs with the web app.

2.2 Backend

There are numerous frameworks for backend available on the market most popular and trusted of which are Django, Spring Boot, and Node Js. Table 2 gives a quick rundown of each of them [4], [50].

Table 2. Overview of backend frameworks

	Node Js	Spring Boot	Django
Initial release	27 May 2009	April 2014	21 July 2005

Programming language	JavaScript	Java	Python
Questions on Stack Overflow	425,360	120,231	288,229
Stars on GitHub	86.7k	60.5k	63.4k
Commits	36,010	37,318	30,635
Contributors	3,109	898	2,198
Advantages	One language for both frontend and backend apps.	Performance. Great framework for developing microservices.	Language simplicity. Quick development process. By default, Django prevents the most common security mistakes: XSS, CSRF, and SQL injection.

Based on the data provided in table 2, Django is the oldest among the aforementioned technologies and has an average number of questions in comparison with Node Js and Spring Boot. But it is important to take into account the fact that Node.js can be searched when developing frontend apps as well, this can significantly affect the number of questions asked. Figure 1 displays the popularity over time metrics of the frameworks during the last 5 years.

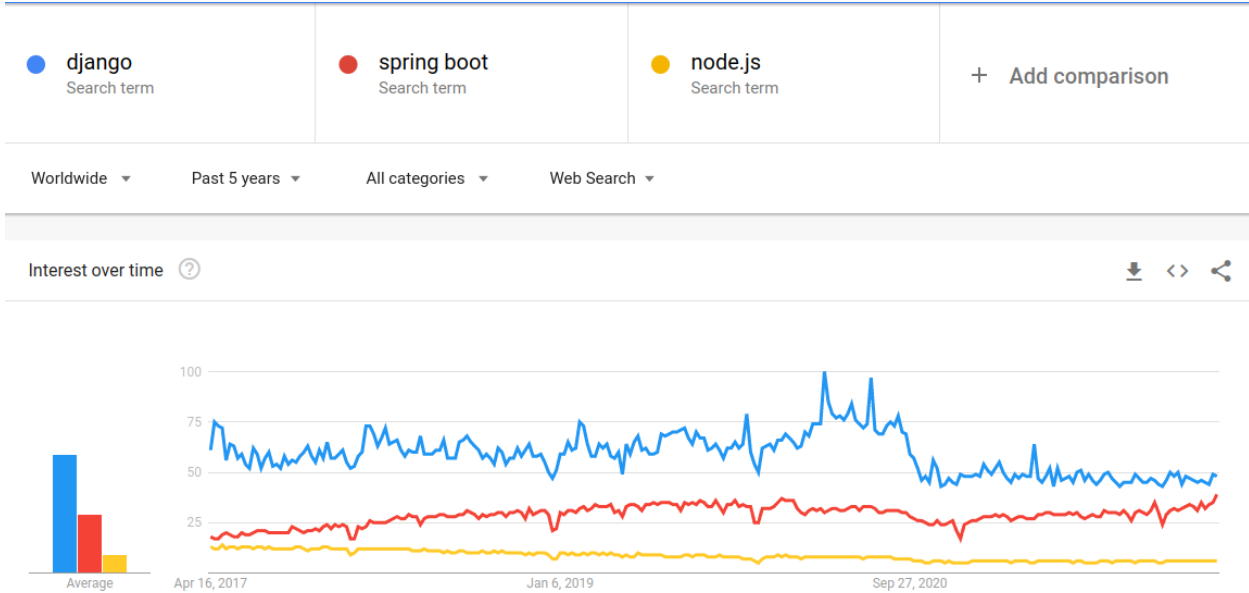


Figure 1. Backend framework popularity according to Google Trends

According to the data provided by Google trends, it is reasonable to conclude that by the year 2022, when the development team started working on the “Self-Developments Camps” Program’s web project, Django was leading the popularity trend 5 years in a row.

Another key consideration is the development speed because the capacity of the development team is limited and the project needs to be delivered in a limited amount of time. That’s why the author has decided in favour of Django.

2.3 Frontend

Some various frameworks and libraries make fronted development faster and easier. Most well-known and trusted are React, Vue.js and Angular. Each of them is open source and used by the world's IT Giants. Table 3 gives a quick rundown of each of them [4], [5], [6], [7], [8].

Table 3. Overview of JavaScript frameworks and libraries

	Angular	Vue.js	React
Initial release	14 September 2016	February 2014	May 29, 2013
Questions on Stack Overflow	275,397	92,189	379,185
Stars on GitHub	80.7k	195k	186k
Commits	24,177	3,226	14,909
Contributors	1,567	404	1,554
Used by	Google, Wix	Alibaba, GitLab	Facebook, Uber
Advantages	Angular allows to write modular services, and have them injected wherever they are	Pre-built components for almost every conceivable use are available from Vue’s large community of	React makes views easier to extend and maintain. By baking an understanding of markup and content into JavaScript, there’s no manual string

	<p>needed. This greatly improves the testability and reusability of the same. Angular is a full-fledged framework, and provides out-of-the-box solutions for server communication, routing within an application, and more.</p>	<p>developers. On top of that, popular libraries exist to provide common functionality such as client-side routing, state management, and server-side rendering!</p>	<p>concatenation and therefore less surface area for XSS vulnerabilities. High performance on working with DOM thanks to reconciliation. In React, you write code that looks like HTML right in your JavaScript code.</p> <p>React supports alternative platforms by breaking diffing and rendering into separate phases. The reconciler computes which parts of a UI have changed, and the renderer uses that information to update the app. This separation means that React Native can use separate renderers to produce iOS and Android apps while sharing the same reconciler, provided by React Core.</p>
--	---	--	---

According to the data from the table, React is the oldest among the aforementioned technologies and it has a bigger number of questions than Vue.js and Angular together on Stack Overflow.

In addition to that, React components can be reused in React Native which is a modern way to build cross-platform mobile applications. Figure 2 displays the popularity over time metrics of the frameworks during the last 5 years worldwide.

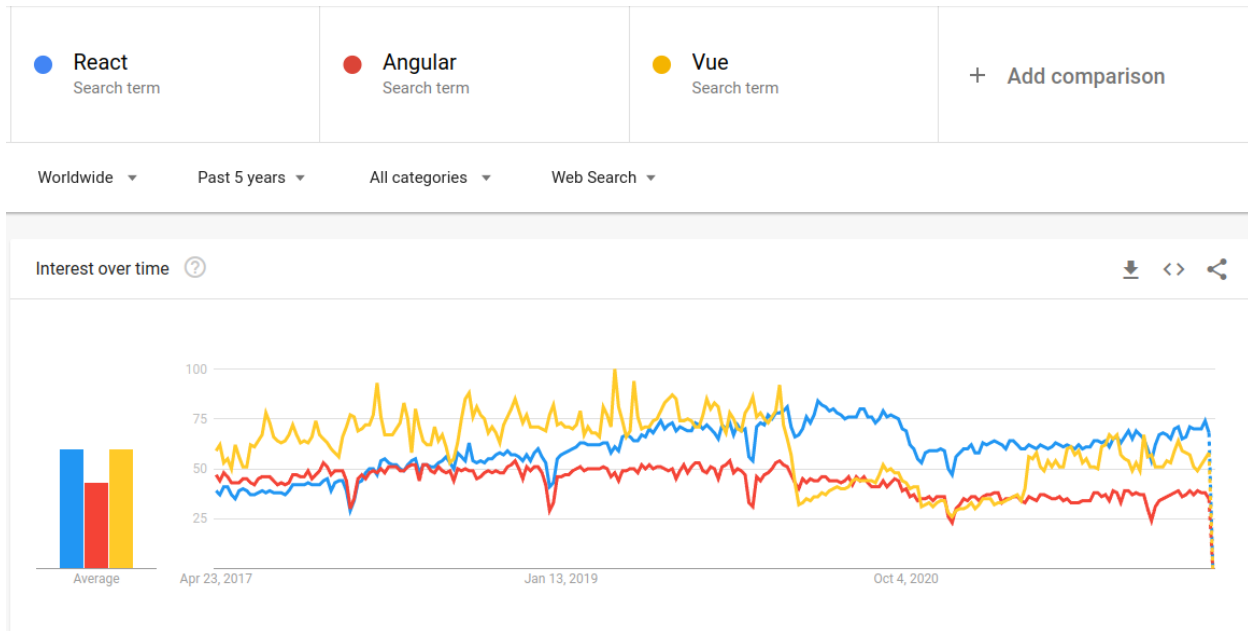


Figure 2. Frontend framework popularity according to Google Trends

Based on the Google Trends chart, it is reasonable to conclude that by the year 2022, when the project began, React was the most popular among the 3 frameworks.

Angular was removed from the candidate technologies list because of its popularity and community size, leaving Vue.js and React as the only options.

As it was stated in paragraph 2.1 (Web development approaches), there is a possibility that in future a mobile application will be developed for the “Self-Development Camps” program. That’s why the author has decided in favour of React.

2.4 Other technologies

In addition to the key stack of Django and React, some more popular nowadays technologies were involved in the project. Each of them is described briefly in the upcoming paragraphs.

2.4.1 Docker and docker-compose

Today installing software is at best inconsistent and overcomplicated. The problem is only worsened if it is needed to make sure that several machines use a consistent set of software over time. Most computers have more than one application installed and running. And most applications have dependencies on other software. Open and closed source programs release security updates continually, and being aware of all the issues is often impossible. The more software to run, the greater the risk that it's vulnerable to attack. These issues can be solved with careful accounting, management of resources, and logistics, but those are mundane and unpleasant things to deal with. The people who built Docker recognised that, and thanks to their hard work, it is possible to breeze through the solutions with minimal effort in almost no time at all [10].

Docker takes away repetitive, mundane configuration tasks and is used throughout the development lifecycle for fast, easy and portable application development – desktop and cloud. Docker's comprehensive end to end platform includes UIs, CLIs, APIs and security that are engineered to work together across the entire application delivery lifecycle [11].

Compose is a tool for defining and running multi-container Docker applications. With Compose, a YAML file is used to configure an application's services. Then, with a single command, it is possible to create and start all the services from the configuration. Compose works in all environments: production, staging, development, testing, and CI workflows [12].

2.4.2 Celery

Celery is an asynchronous task queue/job queue based on distributed message passing. It is focused on real-time operation but supports scheduling as well. The execution units, called tasks, are executed concurrently on a single or more worker server using multiprocessing. Tasks can execute asynchronously (in the background) or synchronously (wait until ready) [55].

Since in the project there was a need to run background tasks to perform complex operations such as parsing and exam checking, the author decided to involve this technology. It is running in a separate Docker container.

2.4.3 Redis

Redis is an in-memory remote database that offers high performance, replication, and a unique data model to produce a platform for solving problems. By supporting five different types of data structures, Redis accommodates a wide variety of problems that can be naturally mapped into what Redis offers, allowing you to solve your problems without having to perform the conceptual gymnastics required by other databases. Additional features like replication, persistence and client-side sharding allow Redis to scale from a convenient way to prototype a system, all the way up to hundreds of gigabytes of data and millions of requests per second [13].

In the product, Dockerized Redis is used to store messages produced by the web application describing the work to be done in the Celery task queue.

2.4.4 PostgreSQL

PostgreSQL is a well-known open-source relational database. The main qualities that attract masses of new users every year and keep current users enthusiastic about their projects are its rock-solid stability, scalability, and safeness, as well as the features that an enterprise-level database management system provides. It has grown to be a whole ecosystem of extensions, tools, and languages tied together by communities spread around the world. PostgreSQL is an open-source project and is fully developed in the open-source world [14]. It is widely considered the most advanced open-source database in the world and provides a wealth of features that are usually only found in commercial databases such as DB2 or Oracle [53].

PostgreSQL has been chosen by the author because it is officially supported by Django, there is an official Docker image for it available on Docker Hub, and it is open-source. It is used as the main database for the web application and is running in a separate Docker container.

2.4.5 Nginx

Nginx is a free, open-source, high-performance HTTP server and reverse proxy. It is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption [51].

Nginx consistently beats Apache and other servers in benchmarks measuring web server performance. Since the original release of Nginx, however, websites have expanded from simple HTML pages to dynamic, multifaceted content. Nginx has grown along with it and now supports all the components of the modern web and the streaming of multiple video formats. Today, Nginx and Nginx Plus can handle hundreds of thousands of concurrent connections and power more of the Internet's busiest sites than any other server. Nginx is the most popular web server on the planet, with more than 350 million websites worldwide relying on Nginx Plus and Nginx Open Source to deliver their content quickly, reliably, and securely [52].

Because of its security, performance, stability, and scalability, the author has decided in favour of Nginx among other available servers. It proxies all the HTTP requests and ensures secure communication.

3 Implementation

The implementation of the proposed solution is discussed in this section in detail. Because the suggested solution is a single page web application, separate client-side and server-side applications are required. Frontend app handles user interface and servers user input. However, the backend is used for fetching and inserting data. Although there is frontend validation for user input, the business level logic is checked on the backend side. The author pays special attention to the security aspects while developing the application.

3.1 IT Architecture

There are 6 main units in the project: Nginx Web Server (reverse proxy), React Single Page Application, Django Restful Web Server, PostgreSQL Database, Redis Message Broker, and Celery Distributed Task Queue Manager. The project is up and running on a Ubuntu machine in the Digital Ocean cloud. Four out of 6 main components are Dockerized and running in separate containers each. They communicate via a docker-compose network. For a better picture, the full architecture is displayed in Figure 3.

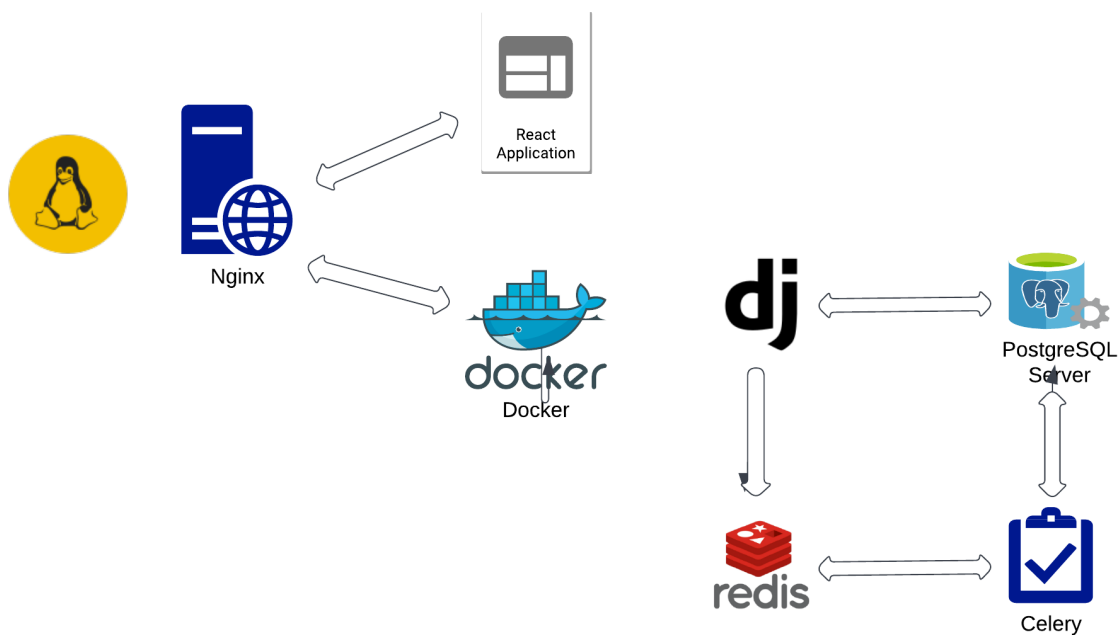


Figure 3. High-level architecture of the application infrastructure (author generated)

3.2 User flow

When a user enters the server IP address or domain name in the search bar, the first service that handles the request is the Nginx webserver. Nginx decides where to proxy the request according to the URL address. If it is not an API request to the Django application then the react application is loaded to the browser and rendered. No more CSS, HTML, or JS is loaded, only data in JSON format is transferred through the network if the user attempts to request data or send something to the backend. Nginx is also responsible for securing traffic between browser and server, using SSL/TLS encryption.

3.3 Database Architecture and Security

Data are at the core of most web applications. Since most operations in user-facing web applications involve data, there is a need to store data in places that are secure, easily accessible, and readily available. A database is a structured collection of data that helps manage information easily. A software layer called the Database Management System is used to store, maintain, and perform operations on the data [20].

However, while developing a web application with multiple tables and fields, SQL commands can easily become overly complex and thus difficult to maintain. For this reason, popular web frameworks such as Django provide a level of abstraction using which we can easily work with databases. The part of Django that helps us do this is called ORM, which stands for Object Relational Mapping. Django ORM converts object-oriented Python code into actual database constructs such as database tables with data type definitions and facilitates all the database operations via simple Python code. Because of this, developers do not have to deal with SQL commands while performing database operations. This helps in faster application development and ease in maintaining the application source code [20]. In addition to that, Django's querysets are protected from SQL injection since their queries are constructed using query

parameterization [21]. Since in the project the database is running in a docker-compose network, it is ensured that only Celery and Django containers can have access to it.

There are many different tables in the databases to support the website's functionality. Figure 4 displays the relation between models in the examination related logic.

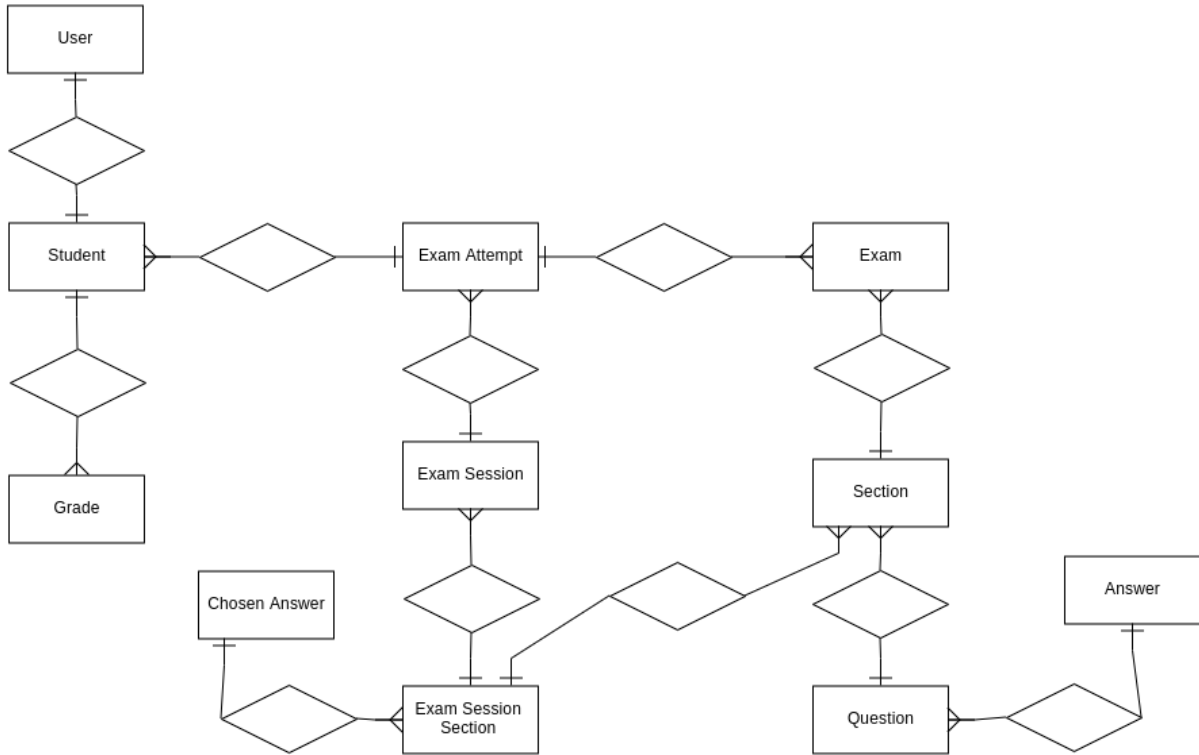


Figure 4. Examination related database scheme (author generated)

3.4 Server-Side App

Backend application is not intractable for end-users. All the data management and business-related operations are controlled here. Since Django projects consist of many apps that provide some set of features [22], it simplifies future development and provides a structured path for additional features.

The server-side is responsible for checking user authority before performing any actions. It is achieved using roles: teacher and student. This way the application is secure from well-known Broken Access Control vulnerability.

There are 7 exploited RESTful endpoints with different operations including listing, fetching, creating, and deleting actions. In the case of a successful response, data is transferred in JSON format. Example response for exam details:

```
{
  "id": 9,
  "exam_date": "2022-04-01",
  "exam_type": "IELTS",
  "participants_count": 5,
  "max_score": 58,
  "participants_with_sections": [],
  "most_common_mistakes": {}
}
```

3.5 Client-Side App

Frontend development is the process of writing HTML, CSS, and JS code with which users will interact in their browser. In this project, React.js was chosen as the main library for developing client-side services.

In React components allow for to reuse of the same structure, and then populate those structures with different sets of data. This adds scalability to the application.

More than 15 functional reusable components with proper routing and input validation were developed for the frontend app, most of them with their module styles. Although checking for user permissions is done on the backend side, it is ensured that some views are only accessible to teachers or students. The components can be used for cross-platform mobile applications using React Native.

4 Security Audit

In this chapter the author makes a security audit for built software using OWASP Top 10 framework, evaluates its results and suggests future improvements.

4.1 Overview of OWASP Top Ten

The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications. It is globally recognized by developers as the first step toward more secure coding. Using the OWASP Top 10 is one of the most effective first steps toward changing the software development culture into one that produces more secure code [27]. Application Security Professionals always keep the OWASP Top 10 as a reference in their career. This top 10 list is always kept up to date by the OWASP community [28].

4.2 Validating security requirements

All of the up-to-date OWASP Top 10 vulnerabilities for the year 2021 are listed below. The author will assess the applicability of each criterion to the developed application and ensure that all applicable requirements are met.

4.2.1 Broken Access Control

Description

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits. Common access control vulnerabilities include bypassing access control checks by modifying the URL, permitting viewing or editing someone else's account, by providing its unique identifier, Accessing API with missing access controls for POST, PUT and DELETE, acting as a user without being logged

in or acting as an admin when logged in as a user [29]. Access control is only effective in trusted server-side code or server-less API, where the attacker cannot modify the access control check or metadata.

Audit

In the developed web application security controls are applied to all the protected endpoints. There are 2 roles only for all the users: student and teacher. Students can see only their attempts while teachers have access to all the exams and exam attempts. Since the application is not yet big, it is not challenging to manage the access control in every view, but access control mechanisms are implemented to be re-used throughout the application for easier and more secure growth. In addition to that, in the frontend application, some components are also only available if the user has a specific role.

Based on the fact that every specific view on the server-side is protected and there is client-side control, the requirements of this vulnerability are deemed as satisfied.

4.2.2 Cryptographic Failures

Description

The first thing is to determine the protection needs of data in transit and at rest. For example, passwords, credit card numbers, health records, personal information, and business secrets require extra protection, mainly if that data falls under privacy laws, e.g., EU's General Data Protection Regulation [30], or regulations, e.g., financial data protection such as PCI DSS [31], [32].

Audit

In the “Self-Development Camps” application, no data is transmitted in cleartext. Only HTTPS is used to communicate between browser and server, HTTP requests are redirected to use a secure connection. For secure communication between client and server RSA algorithm is used. Legacy protocols such as SMTP and FTP are not used at all in the project. Coming to the communication between dockerized containers, they communicate in a single network set up by

docker-compose. Each container for a service joins the default network and is both reachable by other containers on that network, and discoverable by them at a hostname identical to the container name. The only entry point to that network is the Django container which is accessible via binding ports.

Password management is something that should generally not be reinvented unnecessarily, and Django endeavours to provide a secure and flexible set of tools for managing user passwords [33]. In the Django app, the PBKDF2 algorithm with a SHA256 hash (a password stretching mechanism recommended by NIST) is used. It's quite secure, requiring massive amounts of computing time to break. No deprecated cryptographic functions and padding schemes, such as MD5, SHA1, and PKCS number 1 v1.5 are used.

No sensitive data such as credit card numbers, health records, or business secrets are stored. Only secure cryptographic algorithms are used in the project. Hence, the requirements are considered satisfied.

4.2.3 Injection

Description

Cyber-attackers have several vectors for breaking into web applications, but SQL injection continues to be by far their most popular choice. Akamai's State of the Internet report shows that SQL injection now represents nearly two-thirds (65.1%) of all web application attacks [34]. The attack involves entering malicious SQL code into a query that gets executed on the database. It could result in data theft, by dumping database content, or the destruction of data [35].

Audit

Django's database drivers automatically escape the parameters. This ensures that they are treated as pure data and, therefore, they are harmless [35]. Django's querysets are protected from SQL injection since their queries are constructed using query parameterization. A query's SQL code is defined separately from the query's parameters. Since parameters may be user-provided and therefore unsafe, they are escaped by the underlying database driver [21].

There could be instances where there is a need to resort to raw SQL, due to limitations of the Django ORM for example. If a developer is using the low-level ORM API, then he might want to pass bind parameters instead of interpolating the SQL string himself. Even then, it is strongly recommended to check whether each identifier has been properly escaped. But this is not the case in the “Self-Development Camps” web application, no queries are made without usage of Django’s ORM.

No OS commands, and no insecure database queries are made in the implemented app. Hence, the injection requirements are passed.

4.2.4 Insecure Design

Description

Insecure design is a broad category representing different weaknesses, expressed as “missing or ineffective control design”. An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks. One of the factors that contribute to insecure design is the lack of business risk profiling inherent in the software or system being developed, and thus the failure to determine what level of security design is required [36].

Secure design is neither an add-on nor a tool that can be added to the software, it is a culture and methodology that constantly evaluates threats and ensures that code is robustly designed and tested to prevent known attack methods [36].

Audit

The business requirements for the application, including the protection requirements concerning confidentiality, integrity, and availability, were collected and negotiated with the business. Assumptions and conditions for expected and failure flows were analyzed and ensured they are accurate and desirable. Hence, insecure design vulnerability requirements are deemed as satisfied.

4.2.5 Security Misconfiguration

Description

Security misconfiguration vulnerabilities occur when a web application component is susceptible to attack due to a misconfiguration or insecure configuration option. Misconfiguration vulnerabilities are configuration weaknesses that may exist in software components and subsystems or user administration. For example, webserver software may ship with default user accounts that an attacker can use to access the system, or the software may contain sample files, such as configuration files and scripts that an attacker can exploit. In addition, the software may have unneeded services enabled [48].

Audit

A minimal platform without any unnecessary features, components, documentation, and samples is used and there are no default accounts with unchanged passwords. Error handling does not reveal stack traces or other overly informative error messages to users. Segmented application architecture is used to provide effective and secure separation between components.

Development and production environments are configured identically, with different credentials used in each environment.

Deploying the Django app was made according to Django's Official Deployment checklist [49]. Instead of hardcoding the secret values in the code, they are loaded from environment variables. Debug mode is turned off since it enables handy features like full tracebacks in a browser. Nginx redirects all HTTP traffic to HTTPS, and only transmits HTTPS requests to Django.

Since all configurations are secure, requirements are considered satisfied.

4.2.6 Vulnerable and Outdated Components

Description

Software component is part of a system or application that extends the functionality of the application, such as a module, software package, or API. Component-based vulnerabilities occur

when a software component is unsupported, out of date, or vulnerable to a known exploit. For example, an organization may download and use a software component, such as OpenSSL, and fail to regularly update or patch the component as flaws are discovered. Since many software components run with the same privileges as the application itself, any vulnerabilities or flaws in the component can result in a threat to the web application [37].

Audit

In the developed application all the unused dependencies, unnecessary features, components, files, and documentation were removed. Since Docker is used in the project, it is easier to maintain additional packages, because they need to be explicitly listed and installed every time the docker container is rebuilt. No vulnerable, unsupported, or out of date OS, web server, database management system, applications, APIs, runtime environments, or libraries are used. All the used components are only from official sources. Since this type of vulnerability needs special attention, there is an ongoing plan for monitoring, triaging, and applying updates or configuration changes for the lifetime of the application. Therefore, the requirements are met.

4.2.7 Identification and Authentication Failures

Description

Confirmation of the user's identity, authentication, and session management is critical to protect against authentication-related attacks [38]. Identification and authentication failures can occur when functions related to a user's identity, authentication, or session management are not implemented correctly or not adequately protected by an application. Attackers may be able to exploit identification and authentication failures by compromising passwords, keys, and session tokens, or exploit other implementation flaws to assume other users' identities, either temporarily or permanently [39].

Audit

To make sure that users don't use weak or well-known passwords Django's `UserAttributeSimilarityValidator`, `MinimumLengthValidator`, `CommonPasswordValidator`, and `NumericPasswordValidator` validators are used.

For Django RESTful endpoints a server-side and secure Django-Rest-Knox is used for authentication. Knox is an open-source module for Django rest auth that provides easy to use authentication. The module repository has an MIT License, 738 stars, and 369 commits on GitHub. The module aims to allow for common patterns in applications that are REST-based, with little extra effort; and to ensure that connections remain secure. Knox tokens are only stored in an encrypted form. Even if the database were somehow stolen, an attacker would not be able to log in with the stolen credentials [40]. In the "Self-Development Camps" project, the inbuilt mechanism for tokens expiring is set to 10 hours, a default value. Session identifiers are not exposed in the URL and are invalidated after logout.

To confront brute force attacks, Django Rest Framework Throttling [42] is used. It is similar to permissions, in that it determines if a request should be authorized. Throttles indicate a temporary state and control the rate of requests that clients can make to an API. Brute force attempts are logged.

Multifactor authentication is not practical in the case of the Karagandy Bilim-Innovation lyceum since the usage of smartphones is not allowed there.

Since no default credentials are used in the production server, weak password checks are implemented, tokens are saved only in encrypted form, brute force attacks are prevented and logged, and requirements of this type of vulnerability are deemed as satisfied.

4.2.8 Software and Data Integrity Failures

Description

Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or

modules from untrusted sources, repositories, and content delivery networks. An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise. Lastly, many applications now include auto-update functionality, where updates are downloaded without sufficient integrity verification and applied to the previously trusted application. Attackers could potentially upload their updates to be distributed and run on all installations. Another example is where objects or data are encoded or serialized into a structure that an attacker can see and modify is vulnerable to insecure deserialization [41].

Audit

It was ensured that all libraries are consuming from trusted repositories. Since this project is not considered as a higher risk profile, it was considered not to host an internal known-good and vetted repository.

A detailed audit of the used components was made in chapter 4.2.6. It was verified that components do not contain known vulnerabilities.

There is a review process for code and configuration changes to minimize the chance that malicious code or configuration could be introduced into the software pipeline.

At this point, there is no CI/CD pipeline configured to check it for proper segregation, configuration, and access control to ensure the integrity of the code flowing through the build and deploy processes.

Only the JSON format is used in the entire application to send data. DRF serializers are used for that purpose, they allow complex data such as querysets and model instances to be converted to native Python datatypes that can then be easily rendered into JSON, XML or other content types. Serializers also provide deserialization, allowing parsed data to be converted back into complex types, after first validating the incoming data [43].

Only trusted repositories are used, all the components were checked for known vulnerabilities, there is a code review process, no CI/CD is enabled and only DRF serializers are used for validating the incoming data. Hence, the requirements are considered satisfied.

4.2.9 Security Logging and Monitoring Failures

Description

Security logging and monitoring failures are frequently a factor in major security incidents. Failure to sufficient log, monitor, or report security events, such as login attempts, makes suspicious behaviour difficult to detect and significantly raises the likelihood that an attacker can successfully exploit an application [44]. Too much logging bloats the volume of data to oversee, and excessively noisy or disorganized logs make it difficult to glean useful information. On the other hand, sparse logging with insufficient detail might omit critical information, so finding the right balance is an ongoing challenge [45].

Audit

It was ensured that all suspicious login attempts are logged and log data is encoded correctly to prevent injections or attacks on the logging or monitoring systems. In addition to that, all logs are stored in one format so that log management solutions can easily consume those. Celery and Django logs are stored inside the Docker containers and are saved to the production server file system as well using docker-compose volumes [46].

Due to the time and DevSecOps team capacity limits no effective monitoring and alerting system and recovery plan were established to detect and respond quickly to suspicious activities. These are added to the future development plans.

There are no high-value transactions in the application to have an audit trail with integrity controls to prevent tampering or deletion, such as append-only database tables or similar. Proper logging is configured. Therefore, the requirements are passed.

4.2.10 Server-Side Request Forgery

Description

SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list. As modern web applications provide end-users with convenient features, fetching a URL becomes a common scenario. As a result, the incidence of SSRF is increasing. Also, the severity of SSRF is becoming higher due to cloud services and the complexity of architectures [47].

Audit

This vulnerability is indeed applicable to the written software since one of the features there is the parsing of Google Sheets. It could become a problem if someone puts a different link to the input field. However, there is a secure URL parsing before making any requests that sanitize and validate URLs. The algorithm parses the Google Sheet ID from the provided URL and then concatenates it to the Google Spreadsheet URL (<https://docs.google.com/spreadsheet/ccc>). Thus, it is ensured that the requests are only made to the Google API. This is an Application layer protection. Since no other requests are made except to Google, no additional network-layer controls were introduced. Hence, requirements are considered passed.

4.2 Security audit results

As was stated in the problem and background it was agreed to use OWASP Top 10 to check the developed application against security vulnerabilities. This project was used by the development team to check internal procedures for compliance with basic web security criteria.

During the security audit for compliance with OWAS Top 10, the author described requirements and ensured that they are met in the built web application. According to the results, the “Self-Development Camps” web project satisfied all the applicable security requirements,

demonstrating that security aspects of the engineering team's development procedures gave a positive result. No additional security patches were required by the program organizers.

Even though the security audit results were excellent, this may not have been the case for a more feature-rich application. An application that handles credit card numbers, health records, or business secrets would require paying more attention to security guidelines.

5 Further Development

It was also agreed with the program organizers to add more functionality to the web project, but they were not developed in this bachelor diploma work due to time and development capacity limits. The features are listed below in a prioritized order.

5.1 Schedule generator and event management

Creating a schedule take from 5 to 10 hours every week. It is more than one and half months of work for a full-time employee per year. Lack of automation in this process might result in a lower focus of teachers on their main goals as this is a time-consuming activity.

5.2 Teacher cabinet

Now teachers can see detailed attempts of every participant separately. However, it would be much more comfortable to have a separate view for exam reviews where teachers could see all the questions, the answer of each student to each question, the correct answer, and the percentage of correctness among all the exam takers.

5.3 Parent role

At this point, there are only 2 roles in the application: teacher and student. However, it would be beneficial if parents could keep track of their children's progress. Parents shall have similar privileges as students but for each child.

5.4 Automatic recommendations for the program students

English language testing systems consist of 4 main sections: reading, listening, speaking, and writing. It would be useful for students to get a recommendation on which section to focus before taking a real exam.

5.5 Other subjects

At the moment only the English language part is automated. However, it would be profitable to automate other subjects of the program as well.

5.6 Monitoring System

At the current stage there are no effective monitoring and alerting system configured and no recovery plan written for the project. These need to be established to detect and respond quickly to suspicious activities.

6 Conclusion

The goal of the work was to deliver a working web application for the “Self-Development Camps” Program which would help organizers and participants to automate their flow. The project must be accessible online from anywhere using different devices with an internet connection.

The author researched and analyzed various web development methodologies before deciding on Single Page Application approach as the best fit for the project goal. Because of its popularity, language simplicity, development speed, and sensible scalability Django was chosen as the backend framework. For fronted React.js JavaScript library was used due to its popularity and option to easily convert it into a cross-platform mobile application using React Native.

As a result, the author has built an application that matches the program’s functional requirements. The implemented backend API showed to be modular and scalable, which makes it easy to add more applications for user interface as mobile and desktop applications. The React functional components are capable of being used for mobile development.

The author conducted a security audit after the development flow to ensure compliance with the OWASP Top 10 security framework. Following the audit, it was determined that the developed application meets the security requirements, and the project was ready for the deployment phase.

As a final result, the goal of the thesis was achieved, and the website was successfully deployed to Digital Ocean Cloud Service Provider. The project is up and running under the program domain name.

Developments that may be implemented in future to automate other parts of the “Self-Development Camps” Program were listed.

References

1. Emmit A. Scott Jr, “SPA Design and Architecture: Understanding single-page web applications”, 2015.
2. Philip Klauzinski and John Moore, “Mastering JavaScript Single Page Application Development”, 2016.
3. Official Microsoft Documentation. Available at <https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps>. [Accessed 15 April 2022].
4. Motnerail Blog, “Django vs Node.js: When to Choose Which Framework”. Available at <https://www.monterail.com/blog/django-vs-node.js-when-to-choose-which>. [Accessed 15 April 2022].
5. Pete Hunt from the React team, “Why React?”. Available at <https://reactjs.org/blog/2013/06/05/why-react.html>. [Accessed 15 April 2022].
6. Eve Porcello and Alex Banks “Learning React, 2nd Edition”.
7. Nilson Jacques, “Jump Start Vue.js, 2nd Edition”.
8. Shyam Seshadri, “Angular: Up and Running”.
9. React Native home page. Available at <https://reactnative.dev/>. [Accessed 15 April 2022].
10. Jeffrey Nickoloff and Stephen Kuenzli “Docker in Action, Second Edition”.
11. Docker home-page. Available at <https://www.docker.com/>. [Accessed 15 April 2022].
12. Docker documentation. Available at <https://docs.docker.com/compose/>. [Accessed 15 April 2022].
13. Josiah L. Carlson, “Redis in Action”.
14. Luca Ferrari and Enrico Pirozzi, “Learn PostgreSQL”.
16. Adam Freeman, “Pro ASP.NET MVC 5, Fifth Edition”, 2013.
17. Harpreet Singh and Himanshu Sharma “Hands-On Web Penetration Testing with Metasploit”.
18. “About the OWASP Foundation”. Available at <https://owasp.org/about/>. [Accessed 15 April 2022].
19. “Signals in Django”. Available at <https://docs.djangoproject.com/en/4.0/topics/signals/>. [Accessed 15 April 2022].

20. Ben Shaw, Saurabh Badhwar, Andrew Bird, Bharath Chandra K. S., and Chris Guest “Web Development with Django”.
21. “Security in Django”. Available at <<https://docs.djangoproject.com/en/4.0/topics/security/#sql-injection-protection-1>>. [Accessed 15 April 2022].
22. “Applications in Django”. Available at <<https://docs.djangoproject.com/en/4.0/ref/applications/#projects-and-applications>>. [Accessed 15 April 2022].
23. "OWASP top 10 vulnerabilities". developerWorks. IBM. April 20, 2015.
24. “About the OWASP Foundation”. Available at <<https://owasp.org/about/>>. [Accessed 15 April 2022].
25. “OWASP Projects”. Available at <<https://owasp.org/projects/>>. [Accessed 15 April 2022].
26. “OWASP Web Security Testing Guide”. Available at <<https://owasp.org/www-project-web-security-testing-guide/>>. [Accessed 15 April 2022].
27. “OWASP Top Ten”. Available at <<https://owasp.org/www-project-top-ten/>>. [Accessed 15 April 2022].
28. Gus Khawaja. “Practical Web Penetration Testing”.
29. “A01:2021 – Broken Access Control”. Available at <https://owasp.org/Top10/A01_2021-Broken_Access_Control/>. [Accessed 15 April 2022].
30. “What is GDPR, the EU’s new data protection law?”. Available at <<https://gdpr.eu/what-is-gdpr/>>. [Accessed 15 April 2022].
31. Jim Seaman “PCI DSS: An Integrated Data Security Standard Guide”.
32. “A02:2021 – Cryptographic Failures”. Available at <https://owasp.org/Top10/A02_2021-Cryptographic_Failures/>. [Accessed 15 April 2022].
33. “Password management in Django”. Available at <<https://docs.djangoproject.com/en/4.0/topics/auth/passwords/>>. [Accessed 15 April 2022].
34. Ettore Galluccio, Edoardo Caselli, and Gabriele Lombari “SQL Injection Strategies”.
35. Arun Ravindran “Django Design Patterns and Best Practices - Second Edition”.
36. “A04:2021 – Insecure Design”. Available at <https://owasp.org/Top10/A04_2021-Insecure_Design/>. [Accessed 15 April 2022].

37. “Secure against the OWASP Top 10 for 2021| Chapter 6: Vulnerable and outdated components (A6)”. Available at <<https://support.f5.com/csp/article/K17045144>>. [Accessed 15 April 2022].
38. “A07:2021 – Identification and Authentication Failures”. Available at <https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/>. [Accessed 15 April 2022].
39. “Secure against the OWASP Top 10 for 2021 | Chapter 7: Identification and authentication failures (A7)”. Available at <<https://support.f5.com/csp/article/K14998322>>. [Accessed 15 April 2022].
40. “Django-Rest-Knox Documentation”. Available at <<https://james1345.github.io/django-rest-knox/>>. [Accessed 15 April 2022].
41. “A08:2021 – Software and Data Integrity Failures”. Available at <https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/>. [Accessed 15 April 2022].
42. “DRF API Guide. Throttling”. Available at <<https://www.django-rest-framework.org/api-guide/throttling/>>. [Accessed 15 April 2022].
43. “DRF API Guide. Serializers”. Available at <<https://www.django-rest-framework.org/api-guide/serializers/>>. [Accessed 15 April 2022].
44. “Secure against the OWASP Top 10 for 2021 | Chapter 9: Security logging and monitoring failures (A9)”. Available at <<https://support.f5.com/csp/article/K94068935>>. [Accessed 15 April 2022].
45. Loren Kohnfelder “Designing Secure Software”.
46. “Official Docker documentation. Use volumes”. Available at <<https://docs.docker.com/storage/volumes/>>. [Accessed 15 April 2022].
47. “A10:2021 – Server-Side Request Forgery (SSRF)”. Available at <https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/>. [Accessed 15 April 2022].
48. “Secure against the OWASP Top 10 for 2021 | Chapter 5: Security misconfiguration (A5)” Available at <<https://support.f5.com/csp/article/K49812250>>. [Accessed 15 April 2022].

49. “Django Documentation. Deployment checklist”. Available at <https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/>. [Accessed 15 April 2022].
50. Magnus Larsson “Microservices with Spring Boot and Spring Cloud - Second Edition”.
51. Nginx WIKI. Available at <https://www.nginx.com/resources/wiki/>. [Accessed 15 April 2022].
52. Nginx Glossary. Available at <https://www.nginx.com/resources/glossary/nginx/>. [Accessed 15 April 2022].
53. Joshua Drake and John Worsley “Practical PostgreSQL”.
54. “Autocrat Overview”. Available at <https://workspace.google.com/marketplace/app/autocrat/539341275670/>. [Accessed 15 April 2022].
55. “Celery Documentation”. Available at <https://docs.celeryq.dev/en/2.4-archived/getting-started/introduction.html>. [Accessed 15 April 2022].

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis

I Almaz Kydyrmin

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Building a Secure Web Application for the “Self-Development Camps” Program in Kazakhstan.", supervised by Kaido Kikkas

1.1. To be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until the expiry of the term of copyright;

1.2. To be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until the expiry of the term of copyright.

2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.

3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

11.04.2022

Appendix 2 – Screenshots of the application

<Previous 1 Next>

ID	Date	Type	Participants
9	01/04/2022	IELTS	5
10	04/04/2022	IELTS	0
11	04/04/2022	IELTS	0
12	10/04/2022	IELTS	0
13	10/04/2022	YLE	0
14	01/04/2022	IELTS	0

Figure 5. Exam List view for teachers

Exams

Create Exam

Exam Date:	2022-04-01	Total Participants:	5
Exam Type:	IELTS	Maximum Score:	58

Participant	Reading	Listening	Total
Almaz Kydyrmin	29	29	58
Alikhan Khamit	15	15	30
Erkasym Konbay	17	17	34
Dias Baimakanov	20	20	40
Arlan Eljan	12	12	24

Most common mistakes for Reading

Question Number	Incorrect answers
3	5
9	4
14	4
17	4
11	4
7	3
20	3
12	3
6	3
28	3

Most common mistakes for Listening

Question Number	Incorrect answers
3	5
9	4
14	4
17	4
11	4
7	3
20	3
12	3
6	3
28	3

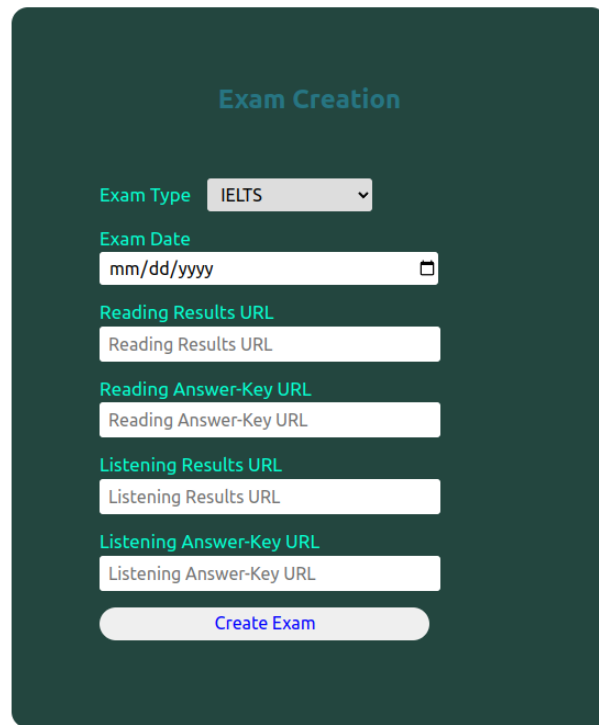
Figure 6. Exam Detail view for teachers

Student [IELTS] [2022-04-18]

Reading

#	Question	Chosen Answer	Correct Answer	?
1	Express train leaves at	B	B	✓
2	Nearest station is	B	A	X
3	Number 706 bus goes to	A	B	X
4	Bus goes to station	C	C	✓
5	Earlier bus leaves at	C	E	X
6	Bus	A	A	✓
7	Train	C	C	✓
8	Tourist	A	A	✓
9	65\$	face	face faces	✓

Figure 7. Exam Attempt Detail view for teachers and students



The image shows a dark green modal window titled "Exam Creation". It contains several input fields: "Exam Type" with a dropdown menu set to "IELTS", "Exam Date" with a text input field containing "mm/dd/yyyy" and a calendar icon, "Reading Results URL", "Reading Answer-Key URL", "Listening Results URL", and "Listening Answer-Key URL", each with a corresponding text input field. At the bottom of the modal is a "Create Exam" button.

Figure 8. Exam Create view for teachers

Upcoming events

<Previous 1 Next>

#	Date	Description
1	09/04/2022 13:00	Camp session #12: IELTS Reading and Listening sections exam
2	18/04/2022 16:00	Camp session #13: IELTS Writing and Reading sections exam
3	25/04/2022 13:00	Camp session #14: SAT Reasoning Mathematics part




Figure 9. Upcoming events view

Contact us

Mobile Phone
+7 775 435 06 36

Email
nursultan.akhmetov@karbil.edu.kz

Social Networks
[t.me/karbil](#) [vk.me/karbil](#) [fb.me/karbil](#)

Ask a question from organizers

Or let us know if you found any website issues

SEND MESSAGE

Figure 10. Contacts and Ask Question view

Self Development Camps Program

"Once you stop learning, you start dying" - Albert Einstein

Nowadays, we see that with the development of technology, everything around us is changing. There are numerous new professions about no one knew on the market today, and some professions, life without which no one could imagine, disappear. The rapid development of technology is changing the way we work, live, and learn. Scientific researches confirm that adaptation and self-development skills are vital skills for the 21st century. Self-study skills and an innovative approach are what make students ready for life's challenges.

In this regard, we, teachers of KarBIL, came to the conclusion that we must focus on developing our students' self-study skills. An annual plan was drawn up for the camps, which take place every three weeks, where special attention is paid to self-education, that is, classes without a teacher. Thus, the student gradually gets used to learning without a teacher and develops self-study skills. Such a format of education has become available relatively recently with the advent of high-quality online educational platforms such as Khanacademy, Busuu, Google Classroom, CodeHS, Quizizz and others.

In addition to classes, during our self-development camps, children go in for sports and participate in various training aimed at preparing them for the future life. Various activities are carried out aimed at improving the relationship between students, educators and teachers. And of course, by reading together we instill a love for books, with the help of which, children learn to be versatile citizens of the 21st century.

It is worth noting that our students and teachers like the new learning approach. And the results are not long in coming. For instance, our lyceum ranks first in terms of the number of books read, is in the TOP-10 in English and in the TOP-5 in mathematics among 38 Bilim Innovation Lyceums in Kazakhstan.

We continue improving our system for organizing camps and automating its elements. At the moment, in coordination with each student, we draw up an individual development plan for him, which will give him the opportunity to be a professional in his field in a rapidly changing modern world.

Organizers BIL teachers

Start Date 17/09/2021

Location Karagandy, Bilim-Innovation Lyceum

Price 70\$

Next camp See the [upcoming events](#)

Subjects English, Math



Figure 11. "Self-Development Camps" Program Description

Student

ID	Date	Type	Total Score	Total Questions
1	18/04/2022	IELTS	52	60

Figure 12. Student Profile View