BAFAR
Architecture Design

Developed by Rayanesh Mobin Hooshyar

Esfand 1403

Technology Stack

- **Main Programming Language:** Python v3.9.x (latest version within 3.9.x series) and Node.js v18.16.x
- **Web Framework:** Django v4.2.x (LTS version) and FastAPI v0.112.x
- **DBMS:** MariaDB v10.4.24
- **Cache and Message Broker:** Redis v7
- **Asynchronous Task Queue and Job Runner:** Celery v5.4.x
- **Periodic Task Schedular:** Celery Beat v2.7.x
- **APM and Issue Tracking System:** Sentry (Self-Hosted) and Sentry (Hamravesh Cloud)
- **Version Control System (VCS):** Git, with GitLab (Self-Hosted) for DevOps lifecycle tool
- **JS Module Bundler:** Webpack v5.x
- **JS Package Manager:** NPM v9.3.x
- **Main JS Library:** jQuery v3.6.x
- **CSS Framework:** Bootstrap v4.6.x
- **Template Engine:** Django Template Engine
- **Object Relational Mapping Tool:** Django ORM

Infrastructure (Production)

- **Containerization Platform:** Docker
- **Orchestration Tool:** Docker Swarm
- **DB High Availability and Horizontal Scaling:** MariaDB MaxScale v2.5
- **Web Server and Reverse Proxy:** Nginx
- **Automation and Configuration Management:** Ansible
- **Continuous Integration/Continuous Deployment (CI/CD):** GitLab CI
- **Automated Database Backup**
- **Automated User Uploaded files Backup:** GlusterFS
- **APM and Issue Tracking:** Sentry

Infrastructure (Stage)

- **Stage Environment:** We have established a stage environment to facilitate the testing and validation of new features and changes. This environment is an exact replica of our production environment, allowing us to:
  - Ensure new features Identify and fix bugs before deployment
  - Function as expected
  - Validate performance and security aspects
- **OS:** Debian 12
- **Containerization Platform:** Docker
- **Orchestration Tool:** Docker Compose
- **Database:** MariaDB v11.4.x
- **Reverse Proxy:** Traefik v2.11.x
- **Web Server:** Nginx
- **Continuous Integration/Continuous Deployment (CI/CD):** GitLab CI
- **APM and Issue Tracking:** Sentry

Security Standards

- **Over HTTPS Connection Only**: Redirect all HTTP requests to HTTPS to ensure encrypted communication.
- **Session-Based Authentication:** Use session-based authentication and transmit the session cookie only over HTTPS (Secure and HttpOnly flags enabled).
- **JWT Authentication (APIs):** Implement rotating refresh tokens and blacklist tokens after rotation to enhance API security and prevent token reuse.
- **Cross Site Request Forgery (CSRF) Protection** & Transmitting the CSRF cookie over HTTPS Connection Only (Secure CSRF Cookie) (Secure and HttpOnly flags enabled)
- **Cross Site Scripting (XSS) Protection** with Automatic HTML Scaping of Django Template Engine
- **SQL Injection Protection** with "Parameterized Queries" and "QuerySet Escaping" of Django ORM
- **Clickjacking Protection** with (X_FRAME_OPTIONS=DENY) or preventing attackers from embedding site in an iframe on a malicious site
- **HTTP Strict Transport Security (HSTS) HTTP Header:** All connections should always use HTTPS, Web Application Protection from man-in-the-middle Attacks
- **Content Security Policy (CSP) HTTP Header:** Detect and mitigate XSS and data injection attacks
- **Permissions-Policy HTTP Header:** Use this header to control browser feature access, such as geolocation, camera, and microphone permissions.
- **Admin Honeypot:** Deploy a fake Django admin login page to trap and log unauthorized access attempts, alerting administrators.
- **User-uploaded file Validation:** Validate uploaded files by checking their file extensions and MIME types to prevent malicious file uploads.

Front-end Features

- **Fully Responsive with Small, Medium and Large Screen Sizes**
- **Dynamic Captcha Method: Google reCAPTCHA or RandomTextCaptchaImage**
  - **Using Google reCAPTCHA in Forms to intelligently avoid automated Requests from bots**
  - **Using RandomTextCaptchaImage to avoid automated Requests from bots (IRAN Access)**
- **Focus on Designing the best UI/UX**
- **Sub Resource Integrity (SRI):** Prevention of unexpected manipulation of resources (or assets) with integrity (hash) tag.
- **Assets Minifying & Chunking:** Removing unnecessary parts from source code without changing its functionality for reduce the size of assets file and also reduce page loading time.
- **Cashe Busting:** Force the browser to load the most recent version of a front-end assets, rather than a previously cached version. We guarantee that the latest version of the front-end assets will always be loaded by the browser.
- **Browser Compatibility:** Supports all browsers that are ES5-compliant with Webpack.
- **Lazy Loading/On-Demand Loading:** Improve the initial loading time of a web page by delaying the loading data until the moment they are actually needed
- **Infinite Scrolling:** Continuously loading more data as the user scrolls down the drop-down. (on some of drop-down, not all of them)
- **Searchable Dropdowns or Dynamic Filtering:** The drop-down list updates in real-time to reflect only the items that match the search query.

Back-end Features

- **Modular Monolith Architecture (Project Level):** Structures the application into independent modules or components with well-defined boundaries. The modules are split based on logical boundaries, grouping together related functionalities. This approach significantly improves the cohesion of the system.
- **Model-View-Controller (MVC) Design Pattern (Module level)**
- **RESTful API & Server-Side Rendering (SSR) Concurrently**
  - **Accepting AJAX Requests** to implement the behavior of Single Page Applications (SPAs)
  - **SSR** can provide faster initial page loads compared to client-side rendering (CSR) alone, especially for users with slower internet connections or less powerful devices. Also, SSR makes the website More SEO friendly by rendering HTML pages on the server side, because many search engine bots do not execute JavaScript.
- **Scalable** (Stateless Architecture)
- **High Available** (Horizontal Scaling and Mirroring)
- **Fast Development, Fast Deployment & Continuous Improvement (Kaizen)**
- **Optimized SQL Queries, Excellent Database Design, Optimal use of indexes and Minimizing the number of executed Queries with Django Object-relational Mapping (ORM)**
- **Following PEP8 (Python Code Specification) Standards**
- **Full-Featured Admin Panel for Support Staff**
- **Support for sending and receiving asynchronous requests for AI modules using Queues and the Producer/Consumer Concept**
- **Support for Asynchronous Processing (Celery & Queues):** Uses Celery for background task execution, improving system performance by offloading heavy computations. Implements the Producer-Consumer pattern with message queues (Redis) to handle long-running or distributed tasks efficiently.
- **Celery Beat for Periodic Tasks:** Schedules and manages periodic tasks. Uses the database to store task schedules, allowing dynamic updates through Django Admin.
- **Using the FallbackCache Mechanism for improved cache reliability ensures that if the primary caching backend fails, the system can seamlessly switch to a secondary cache (Main Memory) preventing data loss and performance degradation.**
- **Using In-memory cache like Redis to reduce database hits and reduce response time.**

Back-end Features

- **Authorization Based On User-Types (10 Different User-Types)**
  - **Natural Users:** Students, Professors, Employees and Admins
  - **Legal Users:** API-Operators, Units, Organizations, Universities, Provinces and Center
- **Using efficient pagination to reducing response time and improving user experience**
- **Detailed Error and Exception Handling, Monitoring and Logging**
- **Secure Login Process with SMS OTPs**
- **Secure Register Process with 2 Layer SMS and Email OTPs for Students**
- **Sending Email Responses to Student Feedbacks**
- **Over <u>30 Different types of Charts</u> with many filters**
- **<u>6 Different</u> types of Cultural-Report including all activities of natural and legal Users**
- **Management and Support <u>5 Different</u> Types of Quizzes with Multiple Questions**
- **Management and Support <u>5 Different</u> Types of Surveys with Multiple Questions**
- **Management and Support Form Builder for Requests with <u>3 Different</u> Types of Inputs**
- **Fully Manageable Events with Multiple Tickets, Multiple Discounts, Grading and Billing**
- **Fully Manageable Elections with 2 Step Confirmation of Candidates and Multi-Vote per User**
- **Management and Support of almost <u>11 Different</u> Types of Organizations with so many features in Membership managements, Events Management, Submitting Self-Declaration, Submitting Requests, Confirming Election Candidates and so on**

Back-end Features

- **Object-Oriented Design (OOD) and Adherence to the Following Principles:**
  - **DRY** (Don’t Repeat Yourself)
    - Improved readability
    - Easier maintenance
    - Faster development
  - **KISS** (Keep It Simple, Stupid), simplicity and clarity in design
    - Simplicity: Simple designs are easier to understand, maintain, and extend
    - Easier maintenance and Faster development
    - Improved Robustness: Simple solutions tend to be more robust because they have fewer components that can fail or interact unexpectedly.
  - **YAGNI** (You Aren't Gonna Need It)
    - YAGNI advises against adding functionality or features until they are actually needed, This principle helps avoid over-engineering and unnecessary complexity.
  - **SoC** (Separation of Concerns) with Modular Monolith Architecture
  - **SOLID Principles**
    - Our design approach has been influenced by SOLID principles
  - **Using some of Design Patterns When needed like Decorator, Factory Method, Simple Factory, and so on.**
- **Continuous Refactoring** to Improve the structure, readability, and maintainability while increasing the number of lines of code and functionalities
- **Continuous Performance Optimization** by profiling and monitoring of system

Statistics

- Over 200,000 lines of developed front-end and back-end code (Medium-scale Project)
- Over 12,600 git commits in almost 2 years
- Almost 18 subsystems including Feedbacks, Quizzes, Surveys, Elections, Events, Requests, Self-Declarations, Organizations and so on, with large amount of functionality.
- Support for more than 420 active Units & 590 Cultural Vice President of Islamic Azad University
- Support for more than 370,000 users including Students, Professors, Employees and Various Type of Operators
- Support for more than 45 Gigabytes Only User-Uploaded files