Back End Development Web-Based TOEIC Test Registration System at the Banyuwangi State Polytechnic Language UPT

Alvin Sugiarto[1], Junaedi Adi Prasetyo[2], Lutfi Hakim[3], Subono[4], Khoirul Umam[5]

[1,2,3,4,5]Software Engineering Technology, Business and Informatics, Banyuwangi State Polytechnic, 68461, Indonesia.

[1]asugiarto60@gmail.com, [2]junaedi.prasetyo@poliwangi.ac.id, [3]lutfi@poliwangi.ac.id, [4]subono@poliwangi.ac.id, [5]khoirulumam@poliwangi.ac.id

***Abstract*— The Language Unit at Banyuwangi State Polytechnic (UPT Bahasa Politeknik Negeri Banyuwangi) is responsible for managing and developing language programs, including English proficiency exams such as the TOEIC. However, the unit faces challenges due to the use of manual methods in registration and participant data management processes. This research aims to develop a web-based TOEIC registration system at UPT Bahasa Banyuwangi State Polytechnic using Firebase as the backend and the Rapid Application Development (RAD) methodology. The system is designed to simplify registration, enhance data management, and improve overall services by providing features like online registration, user and admin dashboards, test schedule management, and certificate issuance. The evaluation of the system shows a successful implementation rate of 98%, with only minor issues related to username or password notifications. Additionally, the system ensures the security of participant data and facilitates easy access to schedule and test result information through a user-friendly dashboard. This development significantly enhances efficiency and transparency in the TOEIC registration process within the educational institution, ultimately leading to better management and service quality for participants and administrators alike. The transition from manual to automated processes not only streamlines operations but also sets a new standard for handling language proficiency exams in educational settings.**

***Keywords*— Development (RAD), Firebase, Rapid Application, React JS, TOEIC.**

I. INTRODUCTION

English has an important role in the world of higher education at Banyuwangi State Polytechnic. To ensure students have adequate English language skills, the Polytechnic requires an English language test or TOEIC through the Language UPT as the organizer of activities starting from registration, language training, and finally the English language test, the results of which will later be used as graduation requirements and graduation requirements. This step not only improves students' English language skills, but also prepares them to compete in a global environment after graduation[1].

The registration and data management process for exam participants is still focused on manual methods, causing a number of significant obstacles, especially in terms of efficiency and accuracy in information management. Some of the obstacles that are clearly visible in the Language UPT at the Banyuwangi State Polytechnic include the high potential for human error in inputting and processing participant data due to manual methods that are prone to negligence and inaccuracy. Second, the administration process is slow and time-consuming, which can have a negative impact on the student experience and cause inconvenience in terms of the smooth process of exam registration. Third, limitations in terms of data analysis and reporting which can hinder quick and appropriate decision making. In the context of innovation and technological progress, these challenges require a more modern and automated approach to improve the quality of UPT Language services and streamline the English language exam registration process.

The integration of Firebase technology as a backend in the development of the English language test registration website at the Banyuwangi State Polytechnic Language UPT aims to overcome obstacles related to the manual system. The main hope is not only to simplify the registration process, but also to increase accuracy and efficiency in managing exam participant data. It is hoped that the success of this implementation will not only provide internal benefits for the Language UPT, but also provide a solid foundation for the transformation towards a more modern language test registration system in the higher education environment. It is also hoped that this can be an example and motivation for other higher education institutions to increase efficiency and effectiveness in the registration process and evaluation of student language skills, thereby strengthening the competitiveness and quality of graduates in the current era of globalization.

This research aims to provide a clear solution in improving the efficiency and accuracy of the English language test registration process at the Banyuwangi State Polytechnic Language UPT. Apart from that, this research also aims to be a driving force in encouraging innovation and improving higher education services globally.

II. METHOD

The website development process uses the Rapid Application Development (RAD) method. RAD is a software

development methodology that emphasizes development speed and active user participation[2]. The main goal is to produce products that can be used immediately through an iterative and incremental process[3].

Additionally, RAD enables developers to identify issues or changes needed early in the development cycle, making it more responsive to user feedback compared to traditional methods like Waterfall[4]. Research has shown that implementing RAD in software development can reduce development time by up to 30% compared to traditional methods, as it emphasizes rapid prototyping and quick iterations[5][6]. Furthermore, RAD promotes better communication between development teams and end-users due to more frequent interactions throughout the development cycle, ultimately enhancing the quality of the final software product[7].

In Figure 1 Rapid Application Development consists of four main phases, namely[2][4]:

1. Planning in which basic needs are identified and feasibility studies are carried out[5]
2. Modeling involves creating and refining prototypes based on user feedback[8]
3. Construction where prototypes are developed into ready-to-use products with a focus on rapid coding and testing[9].
4. Evaluation involves final assessment and implementation of the product, as well as adjustments based on user feedback[9].

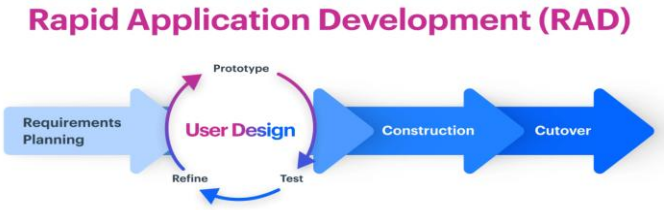


Figure 1. Research flow using the RAD method[10]

Figure 1 is the process of the RAD method flow which consists of the Planning, Modeling, Construction and Evaluation processes. The following research phases will be carried out using the RAD method:

A. Planning

The planning phase in developing the UPT Bahasa Poliwangi website using the RAD method aims to identify user needs and required resources, ensuring all important aspects of the system are well covered. The data required includes user classification, number of registrants per batch, registrant information required by admin, and registration workflow. Data was collected through interviews with the head of the Language UPT who revealed that registration is currently still manual and supports the transition to an online system to increase efficiency, as well as literature studies to get references from trusted journals and sites. Research tool requirements include hardware such as an AMD Ryzen 3 3500 processor and 8 GB DDR 4 3200 MHz RAM, as well as software such as Windows 10 64-bit, React JS, Visual Studio Code, and Firebase.

Functional requirements include participant registration, class selection, user and admin dashboards, and user authentication. Non-functional requirements include responsive website performance, data security, scalability, and suitability for access on various devices. The system to be developed will have a user-friendly interface for online registration, with main features for admins such as test schedule management, uploading result certificates, viewing proof of payment, and deleting user data, as well as features for users such as online registration, user profiles, and downloading. results certificate. This planning phase ensures that all needs and resources have been properly identified, so that the development process can run smoothly and in accordance with the desired goals.

B. Modeling

Modeling involves creating a concept design and initial prototype of the system[11]. This model will continue to be adjusted and improved along with user feedback[12]. Users will be actively involved in model validation and provide constructive feedback[13].

1) Use Cases

The use case depiction in Figure 2 shows the actions that can be taken by the admin and participants in the TOEIC registration system at the Banyuwangi State Polytechnic Language UPT. After logging in, participants are directed to the registrant dashboard to fill out the form, select classes, view the schedule, and download certificates. Admin, after logging in, is directed to the admin page to manage test schedules, upload certificates, view proof of payment, and delete user data. This use case ensures that each role has access and functionality appropriate to their tasks.

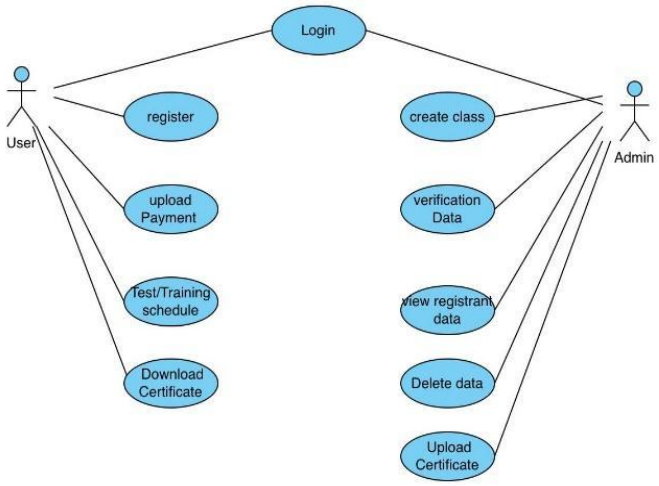


Figure 2. Use case

2) Activity Diagrams

In the activity diagram in Figure 3, the flow of the English test registration system is shown. The flow starts from the user who inputs registration data. After that, the admin manages registrant data, including verifying and processing incoming information. This process continues until the admin uploads the test result certificate. After completing the entire process, participants can download their result certificate. This diagram depicts the main steps in the system, ensuring that all activities from registration to receipt of certificates run smoothly.

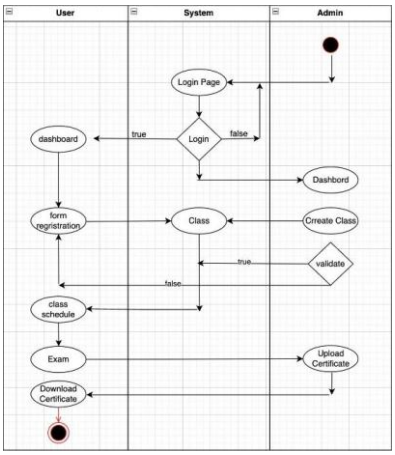


Figure 3. Activity Diagram

3) ER Diagram

ER Diagram is a graphical representation of the structure and relationships between entities in a database[14]. However, when using Firebase Firestore as a NoSQL type database, the relationships between data are not tied into a relationship and its use is more flexible [15]. In Figure 4, for the student data section, the data has been accommodated from the Banyuwangi State Polytechnic SSO API data so that the login process does not require re-registration.

Figure 4. Entity Relationship Diagram (ERD)

C. Implementation

The construction phase involves implementing the prototype that was approved and developed in the previous phase. Users will continue to provide feedback throughout this process, allowing for any changes or adjustments necessary to achieve the optimal solution[16]. During this phase, the development team will focus on coding, data integration, and testing the features present in the prototype to ensure that all functional and non-functional requirements are met[17]. Feedback received from users will be immediately implemented to improve the quality and usability of the system, ensuring that the final product meets the expectations and needs of all stakeholders.

D. Development

Implementation of the project to create an English test registration website using ReactJS begins with selecting the right development environment such as Node.js and NPM for package management. The project structure is organized, including folders for components, pages, and utilities. The initial configuration of the project involves setting up Firebase as a database, including connecting the Firebase API and leveraging dependencies such as the Firebase SDK. The development team then developed a user interface (UI) based on the approved prototype design, implementing components such as a registration form and confirmation page. Functionality such as user registration, exam schedule setting, and admin management is integrated with the Firebase backend using JavaScript logic. Regular testing is performed to ensure system performance and suitability to user needs, including functional and integration testing between the ReactJS front-end and Firebase backend.

E. Evaluation

System evaluation is carried out through thorough testing to ensure optimal performance and quality before full implementation. Users will be involved in testing functionality and providing final feedback. The testing process involves a Table that includes a description of the test, the features tested, success parameters, test results, and a conclusion on the extent to which the test results meet the specified success parameters. The test plan uses a black box to test all the functions that have been implemented.

III. RESULTS AND DISCUSSION

A. System Overview

The TOEIC registration website was developed to facilitate the registration and TOEIC test administration process for prospective participants. This system utilizes Firebase as a backend-as-a-service (BaaS) and React.js as a framework for front end development, providing a good, secure and efficient solution. The main features implemented include online registration, participant data management, and participant certificate management. The login interface of the TOEIC registration system is shown in Fig. 5.

Figure 5. Login Page

B. Back End Architecture

The back end architecture of the proposed system is shown in Fig. 6. The back end architecture uses Firebase for various key functions, including authentication, real-time database, file storage, and cloud functions (Cloud Functions). Firebase's advantage in tight integration with web and mobile platforms makes system development and management easier.

Meanwhile, React.js is used as a front end to build interactive and responsive user interfaces.

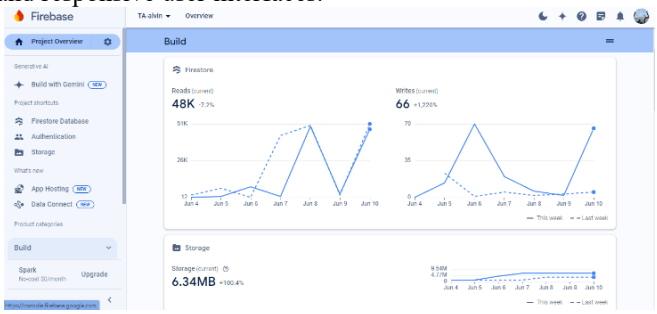
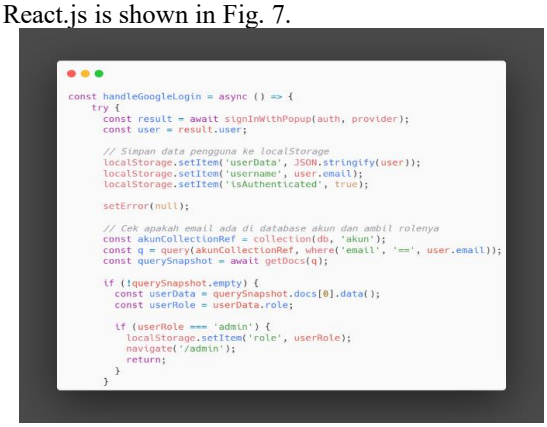


Figure 6. Back End Architecture

C. User Feature Implementation

1) Authentication and Login

User authentication uses Firebase Authentication which supports various login methods, such as email and password, as well as OAuth (Google,). Integration with React.js is carried out to handle the login flow seamlessly. The authentication and login implementation using Firebase Authentication integrated with React.js is shown in Fig. 7.



```
const handleGoogleLogin = async () => {
    try {
      const result = await signInWithPopup(auth, provider);
      const user = result.user;

      // Simpan data pengguna ke localStorage
      localStorage.setItem('userData', JSON.stringify(user));
      localStorage.setItem('username', user.email);
      localStorage.setItem('isAuthenticated', true);

      setError(null);

      // Cek apakah email ada di database akun dan ambil rolenya
      const akunCollectionRef = collection(db, 'akun');
      const q = query(akunCollectionRef, where('email', '==', user.email));
      const querySnapshot = await getDocs(q);

      if (!querySnapshot.empty) {
        const userData = querySnapshot.docs[0].data();
        const userRole = userData.role;

        if (userRole === 'admin') {
          localStorage.setItem('role', userRole);
          navigate('/admin');
          return;
        }
      }
```

Figure 7. Authentication And Login Source Code

The login code applies logic conditioning where the login code will detect the user's role, if you log in using an account that contains the admin role it will be directed to the admin dashboard, if not it will be read as a normal user.

2) Filling in Registration Data

Participant registration data is stored in Firebase Firestore. The registration form is implemented in React JS and the data is sent to Firestore after the submit button is pressed. The source code implementation for storing participant registration data into Firebase Firestore is shown in Fig. 8.

```
<Card.Body style={{ textAlign: 'left' }}>
          <h1 style={{ fontSize: '1.5rem', fontStyle: 'revert' }}>Silahkan Mengisi Form
Pendaftaran</h1>
          <Form>
            <Form.Group className="mb-3" controlId="formName">
              <Form.Label>Nama</Form.Label>
              <Form.Control
                type="text"
                placeholder="Masukkan nama"
                value={newName}
                onChange={(event) => setNewName(event.target.value)}
              />
            </Form.Group>

            <Form.Group className="mb-3" controlId="formPhone">
              <Form.Label>Nomor WhatsApp</Form.Label>
              <Form.Control
                type="text"
                placeholder="Masukkan nomor WhatsApp"
                value={newPhone}
                onChange={(event) => setNewPhone(event.target.value)}
              />
            </Form.Group>
```

Figure 8. Source Code For Filling In Registration Data

3) Download Certificate

On the download certificate page the user is expected to be able to download the certificate if the certificate has been uploaded by the admin. If not, a statement will appear that has not been uploaded. The source code for the certificate download feature is shown in Fig. 9.

```
</Button>
<div className="certificate-container">
  <h2 className="certificate-title">Sertifikat</h2>
  <div className="certificate-list">
    {certificates.map(certificate => (
      <div key={certificate.id} className="certificate-item">
        <p className="certificate-name">{certificate.name}</p>
        {certificate.certificateUrl ? (
          <a href={certificate.certificateUrl} className="certificate-link" target="_blank"
            rel="noopener noreferrer">Lihat Sertifikat</a>
        ) : (
          <span className="certificate-not-uploaded">Belum diunggah</span>
```

Figure 9. Certificate Download Source Code

4) TOEIC Class Management

Logic for checking if classes are full and managing participants using Firestore, with integration in React.js to display class status to users. The implementation of TOEIC class management logic using React.js is shown in Fig. 10.

Full Class Logic in React.js:

```
<strong>Jumlah Pendaftar:</strong> {registrations[kelas.id] || 0}
                </Card.Text>
                {handleCheckQuota(kelas.id, kelas.quota) ? (
                  <div style={{ border: '5px', color: 'red' }}>Kelas Penuh</div>
                ) : verifiedClass && verifiedClass !== kelas.id ? (
                  <div style={{ border: '5px', color: 'red' }}>Anda sudah terdaftar di kelas lain</div>
                ) : (
                  <Button variant="primary" onClick={() => handleEnterClass(kelas)}>Masuk Kelas</Button>
```

Figure 10. Toeic Class Management Source Code

The code logic is to check that if the quota and number of registrants are the same then the class is full and participants cannot enter the class. If the registrants are still the available quota then the class entry button will be active and direct participants to enter the class.

5) User Class Data

Logic for checking if classes are full and managing participants using Firestore, with integration in React.js to display class status to users. The implementation of this logic in the user interface is shown in Fig. 11.

```
{participants.length > 0 ? (
                <Table striped bordered hover>
                  <thead>
                    <tr>
                      <th>Nama</th>
                      <th>Email</th>
                      <th>NIM</th>
                    </tr>
                  </thead>
                  <tbody>
                    {participants.map((participant, index) => (
                      <tr key={index}>
                        <td>{participant.name}</td>
                        <td>{participant.email}</td>
                        <td>{participant.nim}</td>
                      </tr>
                    ))}
                  </tbody>
                </Table>
              ) : (
                <p>Belum ada peserta yang terdaftar di kelas ini.</p>
```

Figure 11. User Class Data Source Code

D. Implementation of Admin Features

1) Admin Dashboard

On the admin dashboard there is some information regarding test registrants, namely the number of registrants, the number of registrants per department, and also statistics on the comparison of average scores between departments. The admin dashboard interface is presented in Fig. 12.

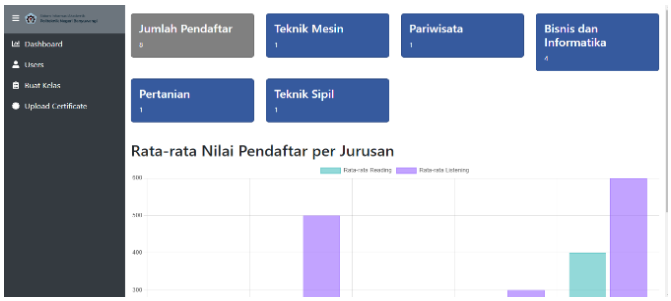


Figure 12. Admin Dashboard Page

2) Participant Data Table

Admin can manage participant data, including sorting and exporting data to Excel. Sorting is done using Firestore queries so admins can filter participant data based on year, class and major. Admins can also search participant name data and export it to Excel using libraries such as SheetJS. The admin interface is developed using React.js to facilitate user interaction. The source code for managing participant data is shown in Fig. 13.

```
const filteredUsers = useMemo(() => {
  const filtered = users.filter((user) =>
    user.name.toLowerCase().includes(searchKeyword.toLowerCase()) &&
    (!selectedJurusan || user.jurusan === selectedJurusan) &&
    (!selectedClass || user.classId === selectedClass) &&
    (!selectedYear || user.registrationYear === selectedYear)
  );

  return filtered;
}, [users, searchKeyword, selectedJurusan, selectedClass, selectedYear]);
```

Figure 13. Source Code For Participant Data Table

3) Class Creation

Admin can create new classes by determining class capacity and schedule. The class creation form is implemented in React.js and the data is sent to Firestore. The implementation of the class creation feature is illustrated in Fig. 14.

```
const handleAddClass = async () => {
    try {
      await addDoc(classesCollectionRef, {
        name: className,
        description: classDescription,
        quota: parseInt(quota),
        schedule: schedule, // Tambahkan field jadwal
      });
      setClassName('');
      setClassDescription('');
      setQuota('');
      setSchedule('');
      setShowModal(false); // Close the modal
    } catch (error) {
      console.error('Error adding class:', error);
    }
  };
```

Figure 14. Source Code For Class Creation

Apart from creating classes, the admin is also equipped with a feature to delete classes and also update quotas to limit the number of registrants on the user side.

4) Upload Certificates and Grades

This feature allows admins to upload participant certificates and scores after the test is completed. The certificate file is uploaded to Firebase Storage and the URL is stored in Firestore. The upload form and data submission logic are implemented in React.js. The source code for uploading certificates and grades is shown in Fig. 15.

```
const handleUploadCertificate = async (user) => {
    if (!user || !certificateFile || isNaN(scores[user.id]?.reading) ||
isNaN(scores[user.id]?.listening)) {
      return;
    }

    setLoading(true);
    setUploadingUser(user.id);
    const storageRef = ref(imageDB, `certificates/${user.id}`);
    const uploadTask = uploadBytesResumable(storageRef, certificateFile);

    try {
      const snapshot = await uploadTask;
      const downloadUrl = await getDownloadURL(snapshot.ref);

      await updateDoc(doc(db, 'pendaftaran', user.id), {
        certificateUrl: downloadUrl,
        readingScore: parseFloat(scores[user.id]?.reading) || 0,
        listeningScore: parseFloat(scores[user.id]?.listening) || 0,
      });

      toast.success('Sertifikat dan nilai berhasil diupload!');
```

Figure 15. Source Code For Uploading Certificates And Grades

E. Functional Testing

After development is complete, thorough testing of the system is carried out to ensure that all functions that have been created can run according to the developer's expectations. The testing plans for users and admins are summarized in Table I and Table II.

TABLE I
USER TESTING PLAN

Test Class	Desired Results	Test result	Information
Login with the correct username & password	Users can enter the user dashboard after entering the correct password and username	According to expectations	Valid
Login with wrong username & correct password	A warning appears that the username/email password is incorrect	Not as expected	Invalid
Login with correct username & incorrect password	A warning appears that the username/email password is incorrect	Not as expected	Invalid
Register while quota is still available	Users can fill in the registration data form after successfully sending and get the information "registered successfully"	According to expectations	Valid
Register when quota is not available	Cannot enter class and a full class warning appears	According to expectations	Valid
Upload proof of payment in png format	Can input image data and send it	According to expectations	Valid
Upload proof of payment in jpg format	Can input image data and send it	According to expectations	Valid
Upload proof of payment other than jpg and png formats	Cannot input image data and sends it with an unsupported format warning	According to expectations	Valid
Upload proof of payment with a file size of more than 1 MB	Cannot input image data and sends it with a warning that the file size is too large	According to expectations	valid
View training and test schedules	Training and test information can be displayed and viewed by the user	According to expectations	Valid
download result certificate	Users can download the resulting certificate file	According to expectations	Valid

TABLE II
ADMIN TESTING PLAN

Test Class	Desired Results	Test result	Infor-mation
Login with the correct username & password	Admin can enter the admin dashboard after entering the correct password and username	According to expectations	Valid
Login with wrong username & correct password	A warning appears that the username/email password is incorrect	Not as expected	Invalid
Login with correct username & incorrect password	A warning appears that the username/email password is incorrect	Not as expected	Invalid
Verify registrant data	The admin can verify the registrant's data whether it is appropriate, if it is not appropriate, the admin can delete the registrant's data	According to expectations	Valid
View participant data and delete data	Admin can view participant data in the participant data Table and can take action to delete participant data	According to expectations	Valid
Upload the results certificate in pdf format	Can input certificate file data to be displayed on the user page	According to expectations	Valid
Upload the results certificate in a format other than PDF	Can input certificate file data and cannot upload it to display on the user page	According to expectations	Valid
Upload the results certificate in PDF format with a file size of 1 MB	Can input and send certificate file data to be displayed on the user page	According to expectations	Valid
Upload the results certificate in PDF format with a file size of more than 1 MB	Unable to input and send certificate file data to be displayed on the user page	According to expectations	Valid
Export Table data to Excel	Admin can export Table data into Excel format files	According to expectations	Valid

From the testing results, it can also be seen in the graphical Table comparing valid and invalid in Figure 16. It shows that there were 2 test cases in user and admin testing that did not work properly, namely logging in with the wrong username or wrong password.

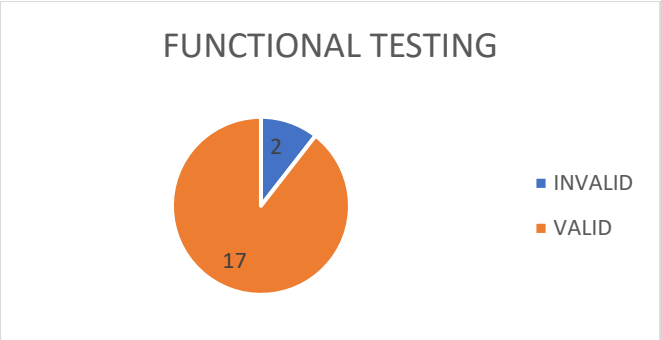


Figure 16. Source code for uploading certificates and grades

IV. CONCLUSION

The implementation of a Firebase-based registration website has effectively addressed the issues of inefficiency and inaccuracy in the manual TOEIC test registration process at the Language Center of Banyuwangi State Polytechnic. By automating the registration process, participants can now register in real-time, significantly reducing the time and effort required for manual data entry. Moreover, the system minimizes input errors, allowing for smoother data management and transparency in participant records. The inclusion of participant and class management features has further enhanced the overall registration experience. Using Firebase as a backend, coupled with React.js for the front end, has improved both efficiency and accuracy in managing participant data. Firebase's secure authentication system ensures safe login processes, while Firestore enables real-time storage and retrieval of participant data. Features such as certificate uploads and class management streamline the handling of critical exam-related tasks, resulting in a faster and more reliable system. This seamless integration between Firebase and React.js has proven essential in managing the TOEIC registration system with improved precision and operational effectiveness.

V. ACKNOWLEDGEMENTS

This research would not have been possible without guidance and support from supervisors and all parties involved at the Banyuwangi State Polytechnic Language UPT. Thank you for the valuable contribution that has been made in the development of the Web-Based TOEIC Test Registration System Back End.

REFERENCES

[1] Nurhalimah, "Optimization Of The English Proficiency Test (Ept) Learning System Using A Digital Platform For State Polytechnic Students." Polytech. Banyuwangi State, p. 63, 2020.

[2] A. Hermanto, "Metode Pengembangan RAD (Rapid Application Development)," 2024. https://agus-hermanto.com/blog/detail/metode-pengembangan-rad-rapid-application-development (accessed Jun. 18, 2024).

[3] D. Hariyanto, R. Sastra, and F. E. Putri, "Implementasi Metode RapidApplication Development Pada Sistem

Informasi Perpustakaan Dicky," J. Al-ilmi, vol. 13, no. 1, pp. 110–117, 2021.

[4] S. M. Fauzi and M. I. Wahyuddin, "Penerapan Rapid Application Development (RAD) Dalam Pengembangan Aplikasi Penjualan Ikan Cupang Hias Berbasis Web," J. Media Inform. Budidarma, vol. 6, no. 2, p. 751, 2022, doi: 10.30865/mib.v6i2.3555.

[5] L. Nilawati, D. Sulastri, and Y. Yuningsih, "Penerapan Model Rapid Application Development Pada Perancangan Sistem Informasi Jasa Pengiriman Barang," Paradig. - J. Komput. dan Inform., vol. 22, no. 2, pp. 197–204, 2020, doi: 10.31294/p.v22i2.8314.

[6] A. Rahman, "Rapid Application Development Sistem Pembelajaran Daring Berbasis Android," Intech, vol. 1, no. 2, pp. 20–25, 2020, doi: 10.54895/intech.v1i2.639.

[7] B. Slivnik, "Context-sensitive parsing for programming languages," J. Comput. Lang., vol. 73, no. 2, 2022, doi: 10.1016/j.cola.2022.101172.

[8] C. Mandang, D. Wuisan, and J. Mandagi, "Penerapan Metode RAD dalam Merancang Aplikasi Web Proyek PLN UIP Sulbagut," Jointer - J. Informatics Eng., vol. 1, no. 02, pp. 49–53, 2020, doi: 10.53682/jointer.v1i02.18.

[9] Lukman Santoso and Juni Amanullah, "Pengembangan Sistem Informasi Akademik Berbasis Website Menggunakan Metode Rapid Application Development (Rad)," Elkom J. Elektron. dan Komput., vol. 15, no. 2, pp. 250–259, 2022, doi: 10.51903/elkom.v15i2.943.

[10] T. Kissflow, "Rapid application development (RAD): Definition, Steps & Full Guide, Kissflow," Kissflow, 2024. https://kissflow.com/application-development/rad/rapid-application-development/ (accessed Sep. 27, 2024).

[11] I. Afriantoro, N. Surojudin, and C. C. Rizkia, "Pelita Teknologi Pengembangan Sistem Informasi Pada Klinik Kenanga Dengan Metode Rad (Rapid Application Development)," J. Pelita Teknol., vol. 16, no. 2, pp. 43–50, 2021.

[12] D. Purnomo, "Model Prototyping," JIMP-Jurnal Inform. Merdeka Pasuruan, vol. 2, no. 2, pp. 54–61, 2017.

[13] D. Y. Descania, "Penerapan Metode Prototype Pada Pengembangan Sistem Antrian Online Di Kementrian Atr/Bpn Kab. Sukabumi," Indexia, vol. 5, no. 01, p. 1, 2023, doi: 10.30587/indexia.v5i01.5165.

[14] K. 'Afiifah, Z. F. Azzahra, and A. D. Anggoro, "Analisis Teknik Entity-Relationship Diagram dalam Perancangan Database Sebuah Literature Review," Intech, vol. 3, no. 2, pp. 18–22, 2022, doi: 10.54895/intech.v3i2.1682.

[15] P. G. Hidayat, N. Suarna, and W. Prihartono, "JITKOM Sistem Informasi Pengelolaan Stok Berbasis Web Menggunakan Metode RUP di Warung Madura Perempatan," J. Ilmu Tek. dan Komput., vol. 08, no. 01, p. 33, 2024, doi: 10.22441/jitkom.v8i1.005.

[16] M. S. P, Muhammad Dedi Irawan, and Ahyat Perdana Utama, "Implementasi RAD (Rapid Aplication Development) dan Uji Black Box pada Administrasi E-Arsip," sudo J. Tek. Inform., vol. 1, no. 2, pp. 60–71, 2022, doi: 10.56211/sudo.v1i2.19.

[17] R. Nuku, E. Masihor, and R. Pasaribu, "Penerapan Metode RAD dalam Perancangan Sistem Informasi Aplikasi Penelusuran Putusan (SIAPP)," Jointer - J. Informatics Eng., vol. 1, no. 02, pp. 54–60, 2020, doi: 10.53682/jointer.v1i02.19.