BAB IV

HASIL DAN PEMBAHASAN

4.1 Implementasi

Tahap ini merupakan proses penerapan sistem secara utuh, baik dari segi perangkat lunak maupun perangkat keras. Tahap dilakukan sesuai dengan proses desain bertujuan untuk menghasilkan sistem yang dibutuhkan pengguna. Bahasa pemrograman yang digunakan adalah bahasa pemrograman java. Berikut pemaparan implementasi.

4.1.1 Implementasi Desain Antarmuka

Tahap yang dilakukan dalam proses implementasi antarmuka disesuaikan pada tahap desain. Hasil dari implementasi antar muka sebagai berikut :

1. **Halaman Awal**

   Saat user pertama menjalankan aplikasi bimbingan. Maka akan muncul gambar seperti yang di bawah ini :

Gambar 4.1 Halaman Awal

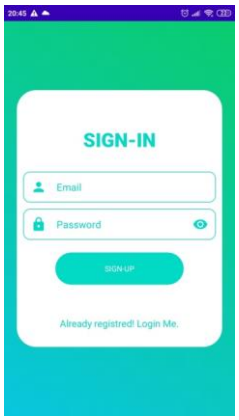
2. Halaman *Login*

Pada halaman ini menampilkan *form login* yang berisikan *email* dan *password* dapat digunakan untuk masuk ke menu utama aplikasi. Halaman *login* ditunjukan pada gambar 4.2.

Gambar 4.2 Halaman *Login*

3. Halaman *Sign Up* Mahasiswa

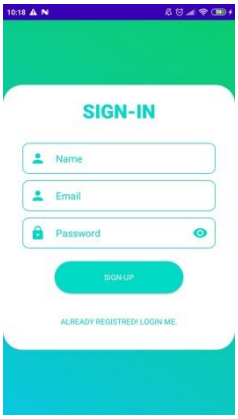
Pada halaman digunakan untuk mendaftarkan akun *user* (Jika belum memiliki akun). Terdapat *email* dan *password* yang harus diisi untuk mendaftar. Halaman Registrasi sepeti gambar 4.3.



Gambar 4.3 Halaman *Sign Up* Mahasiswa

4. Halaman *Sign Up* Dosen

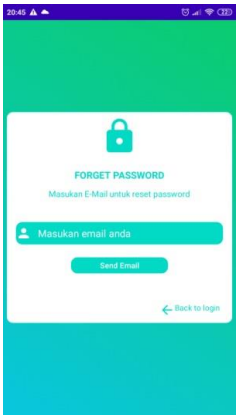
Pada halaman digunakan untuk mendaftarkan akun *user* (Jika belum memiliki akun). Terdapat *name*, *email* dan *password* yang harus diisi untuk mendaftar. Halaman Registrasi sepeti gambar 4.4.



Gambar 4.4 Halaman *Sign Up* Dosen

5. Halaman Lupa *Password*

Pada halaman ini *user* bisa mereset *password* jika *user* lupa. Hanya dengan memasukan *email* yang telah di registrasi setalah itu *email user* akan mendapatkan pesan untuk segera membuat *password* baru. Halaman lupa *password* seperti gambar 4.5.



Gambar 4.5 Halaman Lupa *Password*

6. Halaman Membuat *Profile* Mahasiswa

Pada halaman ini terdapat form *profile* mahasiswa. *User* akan membuat *profile* yang berisikan foto profile, nama, judul skripsi dan jurusan dan memilih dosen pembimbing. Seperti gambar 4.6.

Gambar 4.6 Halaman Membuat *Profile* Mahasiswa

7. Halaman *Resend Email*

Halaman ini untuk mengirim ulang verifikasi email jika *user* tidak mendapatkan sehabis registrasi. Cara menuju halaman ini dengan *user* login terlebih dahulu dan akan ada *box resend email* setelah itu user akan pindah menuju halaman ini. Seperti gambar 4.7.

Gambar 4.7 Halaman *Resend Email*

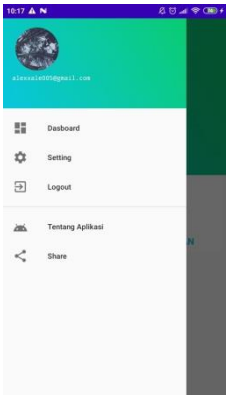
8. Halaman Menu Utama

Pada halaman menu utama ini terdapat box bimbingan yang akan pindah menuju *chat*, *box history* akan menampilkan histori bimbingan mahasiswa. Halaman menu utama pada gambar 4.8.

Gambar 4.8 Halaman Menu Utama

9. Halaman *Navigation* Menu

Halaman ini akan muncul jika *user* mengklik bagian garis tiga di atas pada menu utama. *Navigation* menu berisikan tombol-tombol seperti *dasboard*/ menu utama, *profile*, *logout*, tentang aplikasi. Halaman *navigation* menu pada gambar 4.9.



Gambar 4.9 Halaman *Navigation* Menu

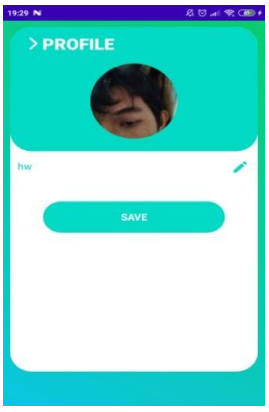
10. Halaman *Profile* Mahasiswa

Halaman ini berisikan foto *profile*, nama, judul dan semester yang telah diisi ketika membuat profile mahasiswa. Disini juga nama, judul dan semester bisa diubah sesuai kemauan *user*/ mahasiswa. Halaman *profile* seperti gambar 4.10.

Gambar 4.10 Halaman *Profile* Mahasiswa

11. Halaman *Profile* Dosen

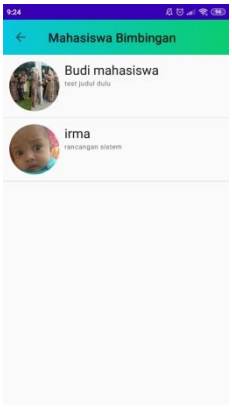
Halaman ini berisikan foto *profile*, nama yang diisi ketika membuat profile. Disini juga nama bisa diubah sesuai kemauan *user* / dosen. Halaman *profile* dosen seperti gambar 4.11.



Gambar 4.11 Halaman *Profile* Dosen

12. Halaman Mahasiswa Bimbingan Dosen

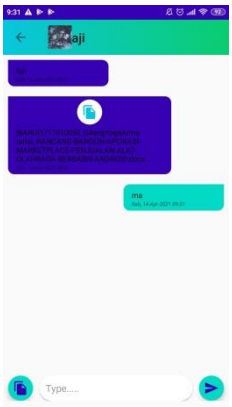
Pada halaman dosen memiliki beberapa mahasiswa bimbingan yang sudah dipilih oleh *user* (Mahasiswa) ketika membuat *profile.* Halaman mahasiswa bimbingan dosen pada gambar 4.12.



Gambar 4.12 Halaman Mahasiswa Bimbingan Dosen

13. Halaman *Chat*

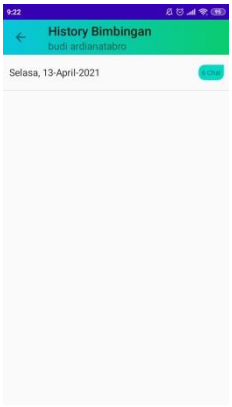
Pada halaman *chat*, *user* bisa berintasksi / bimbingan dengan dosen pembimbing yang telah dipilih ketika membuat *profile* . Halaman *chat* pada gambar 4.13.



Gambar 4.13 Halaman *Chat*

14. Halaman Menu Histori

Pada halaman ini *user* bisa melihat histori ketika bimbingan. Menu histori adalah tempat yang berisikan *chat-chat* dengan dosen pembimbing sesuai dengan hari tanggal dan tahun. Seperti gambar 4.14.



Gambar 4.14 Halaman Menu Histori

15. Halaman Histori Detail

Pada halaman ini *user* bisa melihat histori detail. Halaman ini akan menampilkan *chat-chat* sesuai dengan tanggal bimbingan. Halaman histori detail pada gambar 4.15.

Gambar 4.15 Halaman Histori Detail

4.2. Penulisan Kode Program

Pada bagian ini akan dijeleskan fungsi-fungi setiap halaman yang ada dalam aplikasi beserta kode programnya.

1. **Halaman Awal**

   Pada halaman awal ini berupa *splash screen* yang berisi gambar memuat logo aplikasi. Untuk menampilkannya diatur lama waktu *activity* tersebut muncul di layer. Setelah selesai, maka akan ditampilkan antarmuka selanjutnya.

```
public class MainActivity extends AppCompatActivity {

    Handler handler;
    Runnable runnable;
    ImageView img;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        img = findViewById(R.id.img);
        img.animate().alpha(2000).setDuration(0);

        handler = new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent dsp = new Intent( packageContext: MainActivity.this,Login.class);
                startActivity(dsp);
                finish();
            }
        }, delayMillis: 2000);

    }
    }
```

Gambar 4.16 Kode Program Halaman Awal

Script diatas digunakan untuk menampilkan halaman awal aplikasi yang berupa *splash screen*.

2. Halaman *Login*

Pada halaman ini berupa *form login* yang berisikan *email* dan *password. User* harus menggunkan *email* yang valid dan *password* minimal delapan digit. *Login* digunakan untuk masuk ke halaman menu utama. Kode program sepeti berikut :

```
    if (!TextUtils.isEmpty(email) && !TextUtils.isEmpty(pass)) {
        loginProgress.setTitle("LOGIN GO WISUDA");
        loginProgress.setMessage("Silahkan Tunggu, Akun Sedang Masuk....");
        loginProgress.show();
        mAuth.signInWithEmailAndPassword(email, pass)
                .addOnSuccessListener(authResult -> {
                })
                .addOnFailureListener(e -> Toast.makeText( context: this,  text: "Email Atau Password Salah "
                        + e.getMessage(), Toast.LENGTH_SHORT).show())
                .addOnCompleteListener(task -> loginProgress.dismiss());
    }
}
```

Gambar 4.17 Kode Program *Login*

3. Halaman *Sign Up*

Pada halaman ini *user* yang belum memiliki akun untuk *login.* Akan membuat akun di halaman ini. Disini *user* mengisikan *email* dan *password* yang valid. Maka kode program untuk *sign-in* :

```
    if (!TextUtils.isEmpty(email) && !TextUtils.isEmpty(pass)) {
        mProgress.setTitle("GO-WISUDA");
        mProgress.setMessage("Silahkan Tunggu Sedang Membuat Akun...");
        mProgress.show();
        mProgress.setCanceledOnTouchOutside(false);
        mAuth.createUserWithEmailAndPassword(email, pass)
                .addOnFailureListener(e -> Toast.makeText( context: RegisterActivity.this,  text: "Register gagal "
                        + e.getMessage(), Toast.LENGTH_SHORT).show())
                .addOnCompleteListener(task -> mProgress.dismiss());
    }
}
```

Gambar 4.18 Kode Program *Sign Up*

Kode program di atas menunjukkan *firebaseauth* akan membuat akun *user* setelah sukses email dan password akan masuk ke dalam *database*. Jika register gagal akan mendapatkan pesan.

4. Halaman Lupa *Password*

Pada halaman lupa *password*, *user* bisa mereset kata sandi jika terlupa. Masukan email yang telah terdaftar.

```
Button resetPassword = findViewById(R.id.fgtbutton);
resetPassword.setOnClickListener(v -> {
    String userEmail = inputEmail.getText().toString().trim();
    if (TextUtils.isEmpty(userEmail)) {
        Toast.makeText( context: ForgetPasswordActivity.this,  text: "masukan email", Toast.LENGTH_SHORT).show();
    } else {
        FirebaseAuth.getInstance().sendPasswordResetEmail(userEmail)
                .addOnSuccessListener(aVoid -> {
                    Toast.makeText( context: ForgetPasswordActivity.this,  text: "Reset password terkirim ke email anda",
                            Toast.LENGTH_SHORT).show();
                    startActivity(new Intent( packageContext: ForgetPasswordActivity.this, LoginActivity.class));
                    finish();
                })
                .addOnFailureListener(e -> Toast.makeText( context: this,  text: "Gagal, "
                        + e.getMessage(), Toast.LENGTH_SHORT).show());
    }
});
```

Gambar 4.19 Kode Program Lupa *Password*

Kode program di atas untuk mereset kata sandi baru dengan *firebaseauth* mengirimkan pesan reset sandi ke email *user*.

5. Halaman *Resend Email*

Halaman *resend email* ini berisikan tombol untuk mengirim ulang verifikasi *email* jika email yang telah di registrasi tidak mendapatkannya pesan untuk verifikasi. Berikut ini kode program :

```
        resendEmail.setOnClickListener(view -> sendVerificationEmail());
    }

    private void sendVerificationEmail() {
        FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        if (user != null) {
            user.sendEmailVerification()
                    .addOnSuccessListener(aVoid -> Toast.makeText( context: ResendEmailActivity.this,
                            text: "Verifikasi Sudah Dikirim Ulang, Cek Email anda", Toast.LENGTH_SHORT).show())
                    .addOnFailureListener(e -> Toast.makeText( context: ResendEmailActivity.this,
                            text: "Email Tidak Berhasil Dikirim, " + e.getMessage(), Toast.LENGTH_SHORT).show());
        }
    }
}
```

Gambar 4.20 Kode Program *Resend Email*

6. Halaman Menu Utama

Pada halaman menu utama ini berisikan tombol bimbingan dan *navigation* menu. Tombol bimbingan akan mengarah menuju activity *chat* lalu ada histori dan *navigation* menu *user* hanya menggeser atau mengklik bagian kiri ponsel.

```
navigationView.setNavigationItemSelectedListener(menuItem -> {
    switch (menuItem.getItemId()) {
        case R.id.nav_dasboard:
            startActivity(new Intent( packageContext: this, com.ajikartiko.go_wisuda_dosen.HomeActivity.class));
            finish();
            break;

        case R.id.nav_settting:
            if (currentUser != null) {
                Intent intent = new Intent( packageContext: this, ProfileActivity.class);
                intent.putExtra( name: "CURRENT_USER", currentUser);
                startActivity(intent);
            }
            break;

        case R.id.nav_logout:
            FirebaseAuth.getInstance().signOut();
            break;

        case R.id.nav_android: {
            Toast.makeText( context: this,  text: "TODO FOR CREATE FIEATURE", Toast.LENGTH_SHORT).show();
            break;
        }

        case R.id.nav_share: {
            Intent sharingIntent = new Intent(Intent.ACTION_SEND);
            sharingIntent.setType("text/plain");
            sharingIntent.putExtra(Intent.EXTRA_SUBJECT,  value: "Try now");
            sharingIntent.putExtra(Intent.EXTRA_TEXT, "http://play.google.com/store/apps/detail?id=".concat(getPackageName()));
```

Gambar 4.21 Kode Program *Navigation Menu*

Kode program di atas adalah kode program dari *nagivation* menu terdapat 4 tombol yaitu tombol menuju *acitvity* menu utama, kedua tombol menuju *acitvity profile*, tombol terakhir user logout dan akan menuju *acitvity login.*

7. Halaman *profile*

Pada halaman profile ini *database* akan membaca data yang sudah dibuat *user* pada halaman *profile*. Seperti kode program berikut :

```
public void buttonClickedEdit(View view) {
    if (view.getId() == R.id.buttonEditName) {
        editTextDialog.setTitle("Edit Name");
        editTextDialog.setTextInput( label: "Name", currentUser.getName(),  inputType: null);
        editTextDialog.setListener(result -> {
            currentUser.setName(result);
            nameTextView.setText(result);
        });
        editTextDialog.show(getSupportFragmentManager(),  tag: "buttonEditName");
    } else if (view.getId() == R.id.buttonEditjudul) {
        editTextDialog.setTitle("Edit Judul");
        editTextDialog.setTextInput( label: "Judul", ((Mahasiswa) currentUser).getJudulSkripsi(),  inputType: null);
        editTextDialog.setListener(result -> {
            ((Mahasiswa) currentUser).setJudulSkripsi(result);
            judulTextView.setText(result);
        });
        editTextDialog.show(getSupportFragmentManager(),  tag: "buttonEditJudul");
    } else if (view.getId() == R.id.buttonEditPhoneNo) {
        editTextDialog.setTitle("Edit Semester");
        editTextDialog.setTextInput( label: "Semester", ((Mahasiswa) currentUser).getSemester().toString(), InputType.TYPE_CLASS_NUMBER);
        editTextDialog.setListener(result -> {
            ((Mahasiswa) currentUser).setSemester(Integer.valueOf(result));
            semsterTextView.setText(result);
        });
        editTextDialog.show(getSupportFragmentManager(),  tag: "buttonEditSemester");
    }
}
```

Gambar 4.22 Kode Program *Profile*

Kode program di atas digunakan untuk membaca email user yang telah terdaftar di aplikasi ini.

8. Halaman *Chat*

Pada halaman chat ini berisikan *toolbar* yang terdapat email dan foto *profile*. Seperti code program berikut :

```
private void initToolbar() {
    if (getSupportActionBar() != null) {
        if (otherUser != null) {
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setTitle((TextUtils.isEmpty(otherUser.getName())) ? "chat" : otherUser.getName());
            getSupportActionBar().setLogo(R.drawable.ic_people);
            if (!TextUtils.isEmpty(otherUser.getImage())) {
                RequestOptions requestOptions = new RequestOptions().override( width: 75,  height: 75).diskCacheStrategy(DiskCacheStrategy.ALL);
                Glide.with( activity: ChatActivity.this).asDrawable().apply(requestOptions).load(otherUser.getImage()).into(new CustomTarget<Drawable>() {
                    @Override
                    public void onResourceReady(@NonNull Drawable resource, @Nullable Transition<? super Drawable> transition) {
                        getSupportActionBar().setLogo(resource);
                    }

                    @Override
                    public void onLoadCleared(@Nullable Drawable placeholder) {
                        getSupportActionBar().setLogo(placeholder);
                    }
                });
            } else {
                getSupportActionBar().setLogo(R.drawable.ic_people);
            }
        }
    }
```

Gambar 4.23 Kode Program *Toolbar Profile*

Kode program di atas menunjukan *database* membaca email dan *foto profile* yang ada.

```
        sendButton.setOnClickListener(v -> {
            String text = textInput.getText().toString().trim();
            if (!TextUtils.isEmpty(text)) {
                Message message = new Message(otherUser.getUserId(),
                        FirebaseAuth.getInstance().getCurrentUser().getUid(), Calendar.getInstance().getTime(), text, MessageType.TEXT);
                firestoreUtil.sendMessage(message, currentChatId, task -> {
                    if (task.isSuccessful()) {
                        textInput.setText(null);
                    }
                });
            } else {
                Toast.makeText( context: this,  text: "pesan tidak boleh kosong", Toast.LENGTH_SHORT).show();
            }
        });
        pickFile.setOnClickListener(this::pickFile);
    }
}
```

Gambar 4.24 Kode Program *Send Chat*

Kode program di atas untuk mengirim pesan kepada *user* / dosen ketika berintraksi dan tidak dapat mengirim pesan kosong.

```
if (requestCode == PICK_PDF_FILE && resultCode == Activity.RESULT_OK) {
    Uri uri;
    if (data != null) {
        uri = data.getData();
        ContentResolver contentResolver = this.getContentResolver();
        MimeTypeMap mime = MimeTypeMap.getSingleton();
        String type = mime.getExtensionFromMimeType(contentResolver.getType(uri));
        String displayName = String.valueOf(System.currentTimeMillis()).concat(".").concat(type);
        try (Cursor cursor = contentResolver.query(uri,  projection: null,  selection: null,  selectionArgs: null,  sortOrder: null)) {
            if (cursor != null && cursor.moveToFirst()) {
                displayName = cursor.getString(cursor.getColumnIndex(OpenableColumns.DISPLAY_NAME));
            }
        } catch (Exception e) {
            Toast.makeText( context: this,  text: "Upload Failed " + e.getMessage(), Toast.LENGTH_SHORT).show();
        } finally {
            progressDialog.show();
            StorageReference ref = FirebaseStorage.getInstance() FirebaseStorage
                    .getReference() StorageReference
                    .child(FirebaseAuth.getInstance().getCurrentUser().getUid())
                    .child(displayName.replace( target: "/",  replacement: "-").replace( target: " ",  replacement: "-"));
            ref.putFile(uri).addOnProgressListener(snapshot -> {
                progressDialog.progressBar.setProgress(Math.toIntExact((100 * snapshot.getBytesTransferred() / snapshot.getTotalByteCount())));
                progressDialog.titleDialog.setText("Uploading..");
            }).addOnSuccessListener(taskSnapshot -> {
                Message message = new Message(otherUser.getUserId(), FirebaseAuth.getInstance().getCurrentUser().getUid(),
                        Calendar.getInstance().getTime(), ref.getPath(), MessageType.FILE);
                firestoreUtil.sendMessage(message, currentChatId, task -> {
                    if (task.isSuccessful()) {
                        progressDialog.dismiss();
```

Gambar 4.25 Kode Program *Send File*

Kode program adalah *user* dapat mengirim file berupa pdf/ word jika gagal akan ada pesan " *upload failed* " .

9. Halaman Menu Histori

Pada ini berisikan histori ketika bimbingan dengan *user* (dosen) yang akan dicatat dengan tanggal, bulan dan tahun.

```
private List<History> processHistory(List<Message> messages) {
    ArrayList<History> histories = new ArrayList<>();
    if (messages.size() > 0) {
        Map<String, List<Message>> listMap = messages.stream().collect(Collectors.groupingBy
                (item -> Utils.dateToString(item.getTime(),  pattern: "EEEE, dd-MMMM-yyyy")));

        for (Map.Entry<String, List<Message>> entry : listMap.entrySet()) {
            History history = new History(chatId, entry.getKey(), entry.getValue());
            histories.add(history);
        }
    }
    histories.sort(Collections.reverseOrder());
    return histories;
}
```

Gambar 4.26 Kode Program Histori

Kode program di atas menunjukan list bimbingan / *chat user* yang akan dicatat dengan tanggal, bulan dan tahun.

```
historyAdapter = new HistoryAdapter(item -> {
    if (item instanceof History) {
        Intent intent = new Intent( packageContext: this, HistoryDetailActivity.class);
        intent.putExtra( name: "OTHER_USER", otherUser);
        intent.putExtra( name: "HISTORY_LIST", (History) item);
        startActivity(intent);
    }
});
```

Gambar 4.27 Kode Program Histori Detail

Kode program di atas akan pindah menuju *activity* histori detail yang berisikan chat-chat dengan *user* sesuai tanggal.

4.3. Pengujian Sistem

Pengujian sistem dilakukan dengan cara mencoba fungsi yang telah dibuat pada setiap *activity*. Dalam pengujian sistem aplikasi ini terdapat satu pengujian yaitu user interface.

4.3.1 Pengujian *User Interface*

Pengujian *user interface* bertujuan untuk mengetahui fungsionalitas dari elemen-elemen *interface* yang terdapat di dalam *activity* sistem. Elemen yang diujikan adalah fungsi *button* di setiap aplikasi. Hasil pengujian dapat dilihat pada tabel 4.1.

Tabel.4.1 Tabel Pengujian Sistem

No	Kasus/uji	Skenario uji	Hasil yang didapatkan	Status
1.	Menu Register	Mengisi *username, email,* dan *password* lalu mengklik tombol *register*	Menampilkan halaman *create profile* untuk membuat profile dan menuju halaman *login*	[√] Berhasil [ ] Tidak Berhasil
		Mengosongkan *username, email,* dan *password* ataupun salah satunya lalu mengklik tombol register	Menampilkan pesan pada *email* " masukan email “ dan *password* ” masukan *password* ”	[√] Berhasil [ ] Tidak Berhasil
2.	Menu Lupa *Password*	Mengisi *email* lalu mengklik tombol *send email*	Menampilkan halaman *login* serta menampilkan pesan singkat “*reset password terkirim ke email anda*”	[√] Berhasil [ ] Tidak Berhasil
		Mengosongkan kolom *email* lalu mengklik tombol *reset password*	Gagal dalam proses *reset password* dan menampilkan pesan singkat “error mengirimkan reset *password*”	[√] Berhasil [ ] Tidak Berhasil
3.	Menu *Login*	Mengisi *email* dan *password* lalu mengklik tombol *login*	Menampilkan menu utama	[√] Berhasil [ ] Tidak Berhasil
		Mengosongkan *email* dan *password* ataupun salah satunya lalu mengklik tombol *login.*	Menampilkan pesan pada *email* " masukan email “ dan *password* ” masukan *password* ”	[√] Berhasil [ ] Tidak Berhasil
4.	Menu *Profile*	Melakukan proses mengganti data diri	Berhasil data diri telah berubah	[√] Berhasil [ ] Tidak Berhasil
		Melakukan proses *upload* pada foto.	Menampilkan foto pada menu *profile.*	[√] Berhasil [ ] Tidak Berhasil
5.	Menu bimbingan	Melakukan proses chat dengan dosen	Menampilkan halaman chat	[√] Berhasil

			sebelumnya	[ ] Tidak Berhasil
6.	Menu Histori	Mengklik tombol histori	Menampilan histori *user* yang berisikan tanggal, bulan dan tahun	[√] Berhasil [ ] Tidak Berhasil
		Mengklik tombol histori pada tanggal 6 maret	Menampilkan *chat-chat user* pada tanggal 6 maret	[√] Berhasil [ ] Tidak Berhasil
7.	*Logout*	Mengklik tombol *logout*	Keluar dari menu utama dan menuju halaman *login*	[√] Berhasil [ ] Tidak Berhasil