Project title and short description

This is my project that will be implemented in Android with flask server that accesses data from XAMPP. Digital Badge & Micro-Certification System that does: Earn digital badges for completing key learning tasks, Showcase badges in a personal learning profile, View progress towards certification goals, Share achievements on LinkedIn or student portfolios. The system will be locally hosted using Flask (for backend development) and XAMPP (as the local server environment) to manage databases and APIs efficiently.

Functions that will be implemented in Flask server

1. User Authentication

`register_user()`

- **Method:** `POST`
- **Endpoint:** `/register`
- **Description:** Registers a new user and stores details in the database.
- **Inputs:** `name`, `email`, `password`
- **Returns:** Success or failure message.

`login_user()`

- **Method:** `POST`
- **Endpoint:** `/login`
- **Description:** Authenticates user and returns a session token.
- **Inputs:** `email`, `password`
- **Returns:** Authentication token or error message.

---

2. Profile & User Data

`get_user_profile()`

- **Method:** `GET`
- **Endpoint:** `/profile/<user_id>`
- **Description:** Fetches user details along with earned badges and certification progress.
- **Inputs:** `user_id`
- **Returns:** User profile data.

`update_user_profile()`

- **Method:** `PUT`
- **Endpoint:** `/profile/update`
- **Description:** Allows users to update profile info (name, email, password).

- **Inputs:** `user_id`, `name`, `email`, `password`
- **Returns:** Success or failure message.

---

3. Badge Management

`get_badges()`

- **Method:** `GET`
- **Endpoint:** `/badges`
- **Description:** Retrieves a list of available and earned badges.
- **Returns:** List of badges.

`earn_badge()`

- **Method:** `POST`
- **Endpoint:** `/badges/earn`
- **Description:** Assigns a badge to a user when criteria are met.
- **Inputs:** `user_id`, `badge_id`
- **Returns:** Success message.

`get_badge_details()`

- **Method:** `GET`
- **Endpoint:** `/badges/<badge_id>`
- **Description:** Retrieves badge details (description, criteria, image).
- **Inputs:** `badge_id`
- **Returns:** Badge details.

---

4. Certification Progress

`get_certifications()`

- **Method:** `GET`
- **Endpoint:** `/certifications`
- **Description:** Retrieves a list of available certifications.
- **Returns:** Certification list.

`track_certification_progress()`

- **Method:** `GET`
- **Endpoint:** `/certifications/progress/<user_id>`
- **Description:** Shows a user's progress towards certifications.
- **Inputs:** `user_id`
- **Returns:** Progress data.

`complete_certification_task()`

- **Method:** `POST`

- **Endpoint:** `/certifications/complete_task`
- **Description:** Marks a certification task as completed for a user.
- **Inputs:** `user_id`, `task_id`
- **Returns:** Updated progress.

---

5. Sharing Achievements

`generate_shareable_link()`

- **Method:** `POST`
- **Endpoint:** `/share/badge`
- **Description:** Creates a shareable link for a badge or certificate.
- **Inputs:** `user_id`, `badge_id` or `cert_id`
- **Returns:** URL for sharing.

`share_on_linkedin()`

- **Method:** `POST`
- **Endpoint:** `/share/linkedin`
- **Description:** Integrates LinkedIn API to post achievements.
- **Inputs:** `user_id`, `badge_id`, `message`
- **Returns:** Success message.

---

6. Settings & Miscellaneous

`update_password()`

- **Method:** `PUT`
- **Endpoint:** `/settings/update_password`
- **Description:** Allows users to change their password.
- **Inputs:** `user_id`, `old_password`, `new_password`
- **Returns:** Success or failure message.

`logout_user()`

- **Method:** `POST`
- **Endpoint:** `/logout`
- **Description:** Logs the user out and ends the session.
- **Inputs:** `user_id`
- **Returns:** Success message.

Functions that will be implemented in Android

1. User Authentication

registerUser(name: String, email: String, password: String): LiveData<Response>

- Calls the `/register` API to create a new user.
- Stores the user ID in `SharedPreferences` after successful registration.

loginUser(email: String, password: String): LiveData<Response>

- Calls the `/login` API to authenticate the user.
- Stores authentication token for session handling.

logoutUser(): Boolean

- Clears user session from `SharedPreferences` and redirects to the login screen.

---

2. Profile Management

getUserProfile(userId: Int): LiveData<User>

- Calls `/profile/<user_id>` to fetch the user's profile details, including badges and progress.

updateUserProfile(userId: Int, name: String, email: String, password: String): LiveData<Response>

- Calls `/profile/update` API to update user information.

changePassword(userId: Int, oldPassword: String, newPassword: String): LiveData<Response>

- Calls `/settings/update_password` to update the password.

---

3. Badge Management

fetchBadges(): LiveData<List<Badge>>

- Calls `/badges` API to get all available and earned badges.
- Displays badges in a **RecyclerView**.

earnBadge(userId: Int, badgeId: Int): LiveData<Response>

- Calls `/badges/earn` to assign a badge to a user when criteria are met.

getBadgeDetails(badgeId: Int): LiveData<Badge>

- Calls `/badges/<badge_id>` to fetch details of a specific badge.

---

4. Certification Progress

fetchCertifications(): LiveData<List<Certification>>

- Calls `/certifications` API to retrieve all available certifications.

trackCertificationProgress(userId: Int): LiveData<CertificationProgress>

- Calls `/certifications/progress/<user_id>` to fetch user's progress.

completeCertificationTask(userId: Int, taskId: Int): LiveData<Response>

- Calls `/certifications/complete_task` to update certification progress.

---

5. Sharing Achievements

generateShareableLink(userId: Int, badgeId: Int): LiveData<String>

- Calls `/share/badge` to generate a **shareable link**.

shareOnLinkedIn(userId: Int, badgeId: Int, message: String): LiveData<Response>

- Calls `/share/linkedin` to post badge/certification on **LinkedIn**.

---

6. Local Database (Optional - Using Room)

To provide offline support, use **Room Database** to cache user data.

saveBadgesLocally(badges: List<Badge>)

- Saves fetched badges in **Room Database** for offline access.

getLocalBadges(): LiveData<List<Badge>>

- Fetches badges stored in Room when offline.

Android User Interfaces (you can implement each layout in android, draw in powerpoint, or even draw on a piece of paper and paste here pictures of your design)

1. Splash Screen (`activity_splash.xml`)

- Displays app logo and loading animation.
- Redirects to Login or Dashboard.

2. Login & Registration (`activity_login.xml`, `activity_register.xml`)

- Login page with email/password fields and a login button.
- Registration page with fields for name, email, password, and confirm password.

3. Dashboard (`activity_dashboard.xml`)

- Displays user profile, badge progress, and navigation menu.
- Buttons for "My Badges," "Certifications," and "Share Achievements."

4. Profile Page (`activity_profile.xml`)

- Shows user details (name, email, profile picture).
- Displays earned badges and certification progress.

5. Badge Listing (`activity_badges.xml`)

- Grid/List view of earned and available badges.
- Click on a badge to see details.

6. Badge Details (`activity_badge_details.xml`)

- Badge image, description, criteria for earning.
- “Share” button for LinkedIn/Portfolio.

7. Certification Progress (`activity_certification.xml`)

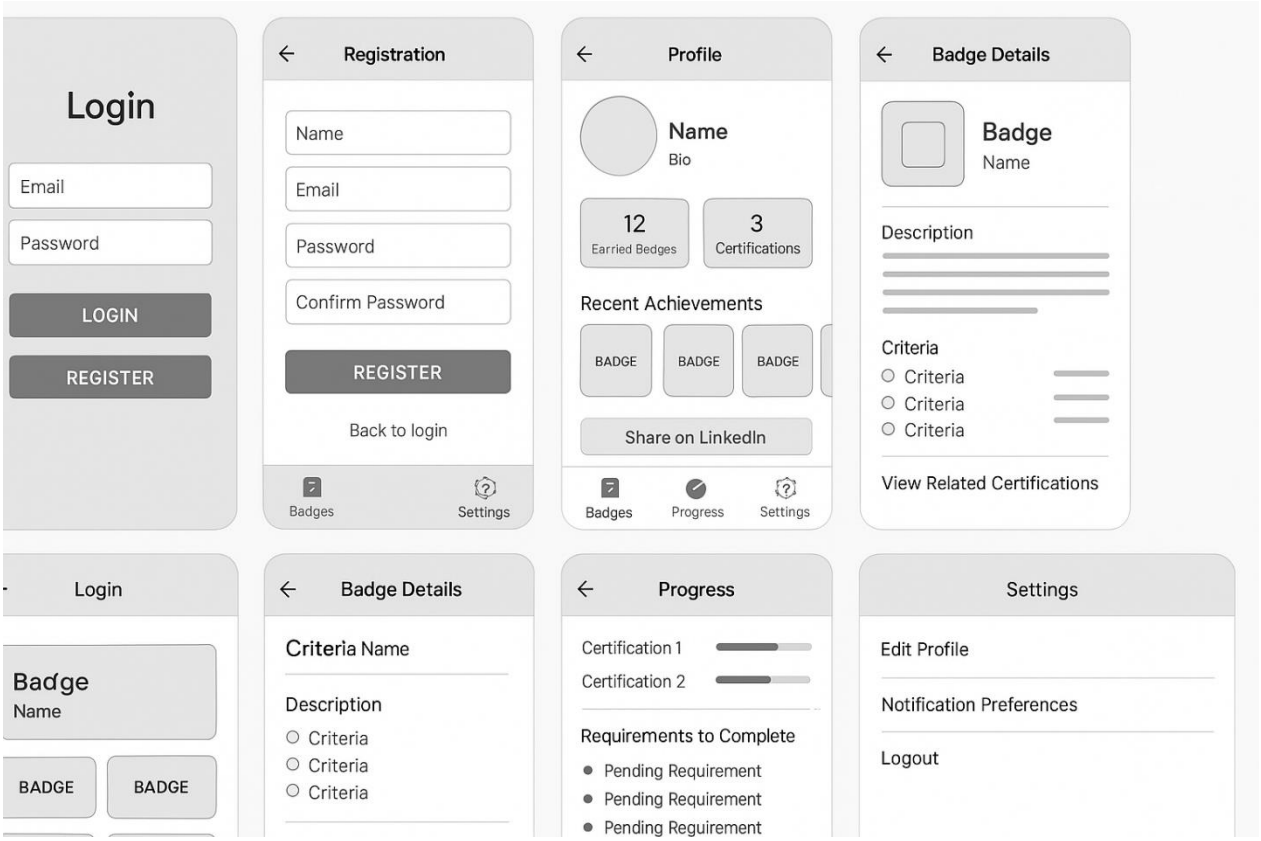
- List of certifications and progress tracking.
- Shows tasks required to earn a certification.

8. Share Achievements (`activity_share.xml`)

- Allows users to generate a shareable badge link.
- Buttons for sharing on LinkedIn or downloading a certificate.

9. Settings (`activity_settings.xml`)

- Options to edit profile, change password, and logout.
-



Describe the flow (goiing) from page to page

(ex: When button create is clicked the data is added to XAMPP table and is displayed on this page)
(ex. When button gotopage2 is clicled, activity that displays page 2 runs and page 2 is displayed)

Flask Project Structure

```
csharp
CopyEdit
digital_badge_system/
├── app/
│   ├── __init__.py
│   ├── models.py          # Database models
│   ├── routes.py          # API routes
│   ├── auth.py            # Authentication handlers
│   ├── badge.py           # Badge management
```

```
│   ├── certification.py # Certification tracking
│   ├── share.py         # Social sharing
├── static/
├── templates/           # HTML templates (if needed)
├── config.py            # Configuration settings
├── run.py               # Main entry point
├── requirements.txt     # Dependencies
├── database.db          # SQLite database (or use MySQL with XAMPP)
```

Database Schema

We'll use **SQLite (for development)** or **MySQL (via XAMPP for production)**.

1. Users Table

```
sql
CopyEdit
CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    password_hash VARCHAR(255),
    profile_pic TEXT
);
```

2. Badges Table

```
sql
CopyEdit
CREATE TABLE badges (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    description TEXT,
    criteria TEXT,
    image TEXT
);
```

3. User_Badges Table (Tracks Earned Badges)

```
sql
CopyEdit
CREATE TABLE user_badges (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    user_id INTEGER,
    badge_id INTEGER,
    earned_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (badge_id) REFERENCES badges(id)
);
```

4. Certifications Table

```
sql
CopyEdit
CREATE TABLE certifications (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    description TEXT,
    required_badges TEXT  -- Comma-separated badge IDs
);
```

5. User_Certifications Table (Tracks Certification Progress)

```
sql
CopyEdit
CREATE TABLE user_certifications (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    user_id INTEGER,
    certification_id INTEGER,
    progress INTEGER DEFAULT 0,  -- Tracks percentage completion
    completed_at TIMESTAMP NULL,
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (certification_id) REFERENCES certifications(id)
);
```

Flask Implementation

1. Setting up Flask (`run.py`)

```
python
CopyEdit
from flask import Flask
from app.routes import app_routes

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'  # Use MySQL
for production
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

app.register_blueprint(app_routes)

if __name__ == '__main__':
    app.run(debug=True)
```

---

2. Models (`models.py`)

```
python
CopyEdit
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(100), unique=True, nullable=False)
    password_hash = db.Column(db.String(255), nullable=False)
    profile_pic = db.Column(db.Text, nullable=True)

class Badge(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    description = db.Column(db.Text, nullable=False)
    criteria = db.Column(db.Text, nullable=False)
    image = db.Column(db.Text, nullable=True)

class UserBadge(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
```

```
    badge_id = db.Column(db.Integer, db.ForeignKey('badge.id'),
nullable=False)
    earned_at = db.Column(db.DateTime, default=db.func.current_timestamp())
```

---

3. API Routes (`routes.py`)

```
python
CopyEdit
from flask import Blueprint, request, jsonify
from app.models import db, User, Badge, UserBadge

app_routes = Blueprint('app_routes', __name__)

# User Registration
@app_routes.route('/register', methods=['POST'])
def register_user():
    data = request.json
    new_user = User(name=data['name'], email=data['email'],
password_hash=data['password'])
    db.session.add(new_user)
    db.session.commit()
    return jsonify({"message": "User registered successfully"}), 201

# Login (Simplified)
@app_routes.route('/login', methods=['POST'])
def login_user():
    data = request.json
    user = User.query.filter_by(email=data['email']).first()
    if user and user.password_hash == data['password']:
        return jsonify({"message": "Login successful", "user_id": user.id})
    return jsonify({"message": "Invalid credentials"}), 401

# Get All Badges
@app_routes.route('/badges', methods=['GET'])
def get_badges():
    badges = Badge.query.all()
    return jsonify([{"id": b.id, "name": b.name, "description": b.description}
for b in badges])

# Earn Badge
@app_routes.route('/badges/earn', methods=['POST'])
def earn_badge():
    data = request.json
    new_earned_badge = UserBadge(user_id=data['user_id'],
badge_id=data['badge_id'])
    db.session.add(new_earned_badge)
    db.session.commit()
    return jsonify({"message": "Badge earned successfully"})
```

Kotlin Implementation (Using Retrofit & ViewModel)

Below is a **Kotlin example** using **Retrofit** to fetch badges.

1. Retrofit API Interface

```
kotlin
CopyEdit
interface ApiService {
```

```
    @POST("register")
    fun registerUser(@Body user: User): Call<Response>

    @POST("login")
    fun loginUser(@Body credentials: LoginRequest): Call<Response>

    @GET("badges")
    fun fetchBadges(): Call<List<Badge>>

    @POST("badges/earn")
    fun earnBadge(@Body badgeRequest: BadgeRequest): Call<Response>
}
```

---

2. ViewModel for API Calls

```
kotlin
CopyEdit
class BadgeViewModel : ViewModel() {
    private val apiService =
RetrofitClient.instance.create(ApiService::class.java)

    fun fetchBadges(): LiveData<List<Badge>> {
        val data = MutableLiveData<List<Badge>>()
        apiService.fetchBadges().enqueue(object : Callback<List<Badge>> {
            override fun onResponse(call: Call<List<Badge>>, response:
Response<List<Badge>>) {
                data.value = response.body()
            }
            override fun onFailure(call: Call<List<Badge>>, t: Throwable) {
                data.value = emptyList()
            }
        })
        return data
    }
}
```

---

3. Fetch & Display Badges in RecyclerView

```
kotlin
CopyEdit
class BadgeActivity : AppCompatActivity() {
    private lateinit var badgeViewModel: BadgeViewModel

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_badges)

        badgeViewModel =
ViewModelProvider(this).get(BadgeViewModel::class.java)

        badgeViewModel.fetchBadges().observe(this, Observer { badges ->
            recyclerView.adapter = BadgeAdapter(badges)
        })
    }
}
```

Android UI Layouts:

```
/*
```

```
1 Login & Registration (LoginActivity.kt & activity_login.xml)
*/

// LoginActivity.kt
class LoginActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        val btnLogin = findViewById<Button>(R.id.btnLogin)
        btnLogin.setOnClickListener {
            val email = findViewById<EditText>(R.id.etEmail).text.toString()
            val password = findViewById<EditText>(R.id.etPassword).text.toString()
            loginUser(email, password)
        }
    }

    private fun loginUser(email: String, password: String) {
        // API call using Retrofit
    }
}

/* activity_login.xml */
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <EditText android:id="@+id/etEmail" android:hint="Email" />
    <EditText android:id="@+id/etPassword" android:hint="Password" android:inputType="textPassword"
/>
    <Button android:id="@+id/btnLogin" android:text="Login" />
</LinearLayout>

/*
2 Dashboard (DashboardActivity.kt & activity_dashboard.xml)
*/

// DashboardActivity.kt
class DashboardActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_dashboard)
    }
}
```

```
/* activity_dashboard.xml */
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView android:text="Welcome to Dashboard" android:textSize="20sp" />
    <Button android:text="View Badges" android:onClick="goToBadges" />
</LinearLayout>

/*
3 Badge List (RecyclerView UI) (BadgeActivity.kt & activity_badges.xml)
*/

// BadgeActivity.kt
class BadgeActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_badges)
    }
}

/* activity_badges.xml */
<?xml version="1.0" encoding="utf-8"?>
<RecyclerView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/recyclerViewBadges"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

/*
4 Badge Details Page (BadgeDetailsActivity.kt & activity_badge_details.xml)
*/

// BadgeDetailsActivity.kt
class BadgeDetailsActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_badge_details)
    }
}

/* activity_badge_details.xml */
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
   android:orientation="vertical"
   android:padding="16dp">

   <ImageView android:id="@+id/badgeImage" android:layout_width="100dp"
android:layout_height="100dp" />
   <TextView android:id="@+id/badgeName" android:textSize="18sp" />
   <TextView android:id="@+id/badgeDescription" android:textSize="16sp" />
</LinearLayout>

/*
5 Certification Progress Page (CertificationActivity.kt & activity_certification.xml)
*/

// CertificationActivity.kt
class CertificationActivity : AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_certification)
   }
}

/* activity_certification.xml */
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:padding="16dp">

   <TextView android:text="Certification Progress" android:textSize="20sp" />
   <ProgressBar android:id="@+id/certificationProgress"
style="?android:attr/progressBarStyleHorizontal" />
</LinearLayout>

/*
6 Profile Page (ProfileActivity.kt & activity_profile.xml)
*/

// ProfileActivity.kt
class ProfileActivity : AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_profile)
   }
}

/* activity_profile.xml */
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:padding="16dp">

   <ImageView android:id="@+id/profilePic" android:layout_width="100dp"
android:layout_height="100dp" />
   <TextView android:id="@+id/profileName" android:textSize="18sp" />
</LinearLayout>

/*
7 Share Badge Page (ShareBadgeActivity.kt & activity_share_badge.xml)
*/

// ShareBadgeActivity.kt
class ShareBadgeActivity : AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_share_badge)
   }
}

/* activity_share_badge.xml */
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:padding="16dp">

   <TextView android:text="Share Badge on LinkedIn" android:textSize="20sp" />
   <Button android:text="Share" android:onClick="shareBadge" />
</LinearLayout>
```