TUNIS BUSINESS SCHOOL

TuLink

Connecting Expertise, Empowering Startups

An API Platform for the Tunisian Startup Ecosystem

Author

Ibtihel Idoudi

BA/IT

Supervisor

PR. Montassar Ben Messaoud

January 2025

DECLARATION OF AUTHORSHIP

We declare on our honour that the work presented in this dissertation, entitled "TuLink - An API platform Connecting Tunisian Startups with Experts," is original and was carried out by **Ibtihel Idoudi** under the supervision of **Professor Montassar Ben Messaoud**.

January 2025

ABSTRACT

This project addresses the critical need for improved access to expert guidance within the Tunisian startup ecosystem. **TuLink**, a unified API platform, is introduced to provide a seamless, efficient solution. This API directly connects Tunisian startups with local experts, offering features such as streamlined matchmaking, booking, and consultation management. The goal is to expand access to essential expertise, enhance collaboration, and accelerate the growth of the Tunisian startup ecosystem. The platform implements authentication, booking, and payment mechanisms to ensure a secure and user-friendly experience.

Keywords: API Platform, Tunisian Startups, Expertise Marketplace, Matchmaking, Booking System.

Contents

List of Figures

1. INTRODUCTION

The Tunisian startup ecosystem, though a growing source of innovation and economic activity, faces significant challenges in connecting with the necessary expertise required for its growth. This ecosystem, characterized by a diverse range of startups with unique needs, struggles due to limited access to tailored, timely, and affordable guidance. Startups navigate a fragmented landscape, encountering hurdles in the areas of complex matchmaking, inefficient scheduling, and disparate communication, all of which hinder their progress and create unnecessary friction.

To address these hurdles, this project proposes **TuLink**, an innovative API-driven platform designed to revolutionize how Tunisian startups connect with expert consultants. Acting as a centralized hub, **TuLink** streamlines the process of finding and engaging with diverse experts across various fields, enabling a more efficient and targeted approach to seeking specialized knowledge and skills. It aims to provide an integrated ecosystem where startups can discover qualified consultants, schedule consultations, and manage communication seamlessly. This platform is designed to bring value for both startups and consultants.

This project focuses on the design, and development of this unified API platform, delving into its technical architecture and core functionalities. It also explores the competitive advantages and innovative approaches of **TuLink**, as well as its potential impact on the Tunisian startup ecosystem. The following sections of this report will present a detailed overview of the platform, its features, and its expected contribution to creating a more connected and innovative environment in Tunisia.

2. MOTIVATION

2.1 Problem Statement

Tunisian startups often face difficulties in finding timely and affordable expertise to address the specific challenges they encounter during their growth phase. Existing solutions typically lack personalized matchmaking and integrated scheduling or communication capabilities. This leads startups to navigate fragmented resources and inefficient processes. This fragmentation leads to:

- Limited Access: Difficulty in finding suitable experts in various fields.
- Inefficient Processes: Lack of integrated scheduling and communication tools, creating additional overhead and delays.
- High Costs: Existing solutions may be expensive, hindering accessibility for smaller startups.

2.2 Solution

TuLink aims to revolutionize the Tunisian startup ecosystem by offering a unified API platform that enables seamless connections between startups and experts across diverse domains. This platform provides a structured and user-friendly environment for matchmaking, booking, and managing online consultations, and seeks to deliver specific and substantial benefits to both startups and experts:

- **Startups:**
  - **Targeted Matchmaking: TuLink** provides a system that uses specific parameters and algorithms to match startups with relevant experts efficiently.
  - **Streamlined Booking & Communication:** Integrated tools allow startups to easily view availability, request bookings, reschedule, and communicate directly through chat or video calls.
  - **Cost-Effective Expertise:** Startups can access a range of experts, compare prices and choose the most affordable options, with clear and transparent pricing.
  - **Centralized Management:** Startups can manage their interactions with experts, track bookings, and review consultation history from a single dashboard.
- **Experts:**
  - **Showcase Expertise:** Experts can create detailed profiles, share their port-

folio, and highlight their specializations to attract relevant startups.

- **Flexible Scheduling:** Experts have control over their availability, managing bookings and consultations efficiently.
- **Wider Network: TuLink** provides access to a broad network of Tunisian startups seeking their specific expertise.
- **Efficient Consultation Management:** Integrated tools enable streamlined management of bookings, payments, and communications.

3. COMPETITIVE ANALYSIS

While a direct, unified API platform connecting startups with experts like **TuLink** may not exist yet in Tunisia, it's essential to position it against popular alternatives that startups may use to find expertise. We will analyze **TuLink**'s competitive edge with respect to LinkedIn and freelancing platforms.

3.1 Comparison with LinkedIn

LinkedIn serves primarily as a professional networking and career platform. While it allows startups to search for experts and potentially connect with them, it has significant limitations in the context of targeted, project-based collaboration:

- **Lack of Focused Matchmaking:** LinkedIn's search functionality is broad and not optimized for matching startups with specific expertise. Startups might have to sift through numerous profiles that are not relevant to their needs.
- **Absence of Booking Systems:** There's no built-in system for booking consultations, scheduling calls, or tracking engagement on LinkedIn. Communication happens via messages or external means.
- **Limited Transparency and Pricing:** LinkedIn profiles do not typically detail pricing, past collaborations, ratings and reviews, making it harder for startups to make informed decisions.
- **No Structured Project Management:** LinkedIn lacks dedicated project management features, making it difficult to manage a consulting engagement efficiently.

TuLink addresses all of these issues with its targeted matchmaking, integrated booking and scheduling tools, and structured project management features.

3.2 Comparison with Freelancing Platforms

Freelancing platforms like Upwork or Fiverr provide access to a pool of global freelancers. While they offer a structured environment for hiring and managing projects, they present the following challenges for Tunisian startups:

- **Lack of Localized Expertise:** Freelancing platforms offer a global marketplace, where most experts are not Tunisian and lack localized knowledge and experience, which can be essential for Tunisian startups.
- **Pricing and Payment Issues:** Global platforms have complicated payment sys-

tems, pricing may not be affordable for all Tunisian startups, and currency exchange can be an obstacle.

- **Generic Talent Pool:** Freelancing platforms can be overwhelming with the sheer number of freelancers and profiles.
- **Lack of Integration:** Freelancing platforms often lack specialized communication and project management tools making it difficult to manage project effectively.

TuLink is designed to address these issues by building a platform that focuses on a local network of experts, integrated communication and project management tools to enable better collaboration between the experts and the startups.

3.3 TuLink's Competitive Advantage

TuLink's competitive advantage lies in its tailored approach that's built specifically for the Tunisian market. Here are some key advantages of this API:

- **Targeted Matchmaking**: **TuLink** provides a specialized system for connecting Tunisian startups with local experts, making the discovery process efficient and highly relevant.
- **Integrated Booking and Scheduling**: The platform's built-in tools for scheduling consultations, managing bookings, and tracking engagement streamline workflows for both parties.
- **Transparent and Secure Ecosystem**: **TuLink** encourages transparency through ratings, reviews, and clear pricing systems. In addition to that, the payment methods are adapted to the local market and with secure payment features.
- **Project Management Tools**: **TuLink** offers built-in tools that are specifically designed for managing consulting engagements effectively.
- **Local Network**: **TuLink** is designed to foster a network of local Tunisian experts, offering specialized experience that is relevant to the Tunisian market.

By focusing on the specific needs of the Tunisian market, **TuLink** aims to provide a far more efficient, transparent, and user-friendly experience than generic alternatives, accelerating growth and innovation in the Tunisian startup ecosystem.

4. TECHNOLOGIES USED

This section details the key technologies and tools employed throughout the development, deployment, and operational phases of the **TuLink** API platform. These technologies were selected for their robustness, performance, and suitability for the project's objectives.

4.1 FastAPI

Description: FastAPI is a modern, high-performance Python framework serving as the API's core. It provides a robust structure for handling requests, routing, and data serialization with speed and ease. Its asynchronous capabilities enable the API to handle a high number of requests efficiently.

4.2 FastAPI Swagger UI

Description: An interactive API documentation interface automatically generated by FastAPI. This tool allows developers to visualize endpoints, understand parameters, and test the API directly in the browser, making API exploration and debugging much easier.

4.3 Postman

Description: A tool used for API testing and debugging, Postman enables creation, sending, and analyzing of HTTP requests. It helps validate the API's behavior and ensure consistent functionality.

4.4 MySQL Workbench

Description: MySQL Workbench is a visual database design tool, which allows us to create database schemas and tables in an efficient way. It was used to design and create the database that will be used by the **TuLink** API.

4.5 SQLAlchemy

Description: A Python SQL toolkit and ORM used for interacting with the database. It provides an abstraction layer, allowing us to interact using Python objects, leading to cleaner, more maintainable, and more secure code.

4.6 JSON Web Tokens (JWT)

Description: JWTs are a standard method for securely sending and validating data, primarily used for user authentication. JWTs enable stateless communication by encoding user information in tokens.

5. SYSTEM ARCHITECTURE AND DESIGN

This section outlines the database design and API structure of the **TuLink** platform.

5.1 Database Usage

The **TuLink** API platform manages its data using a MySQL database. The database schema is created and managed through MySQL Workbench and SQLAlchemy ORM. Data models are implemented using the SQLAlchemy declarative base to ensure that data types are properly assigned.

5.2 Data Models

The following data models represent the core objects of the **TuLink API Platform**, with their attributes and relationships. These models help in defining the structure and data handling within your platform:

- **User**: Represents a user account within the **TuLink** platform, capable of acting as a startup representative, an expert, or an admin.
- **Admin**: Represents an administrator account within the **TuLink** platform.
- **ServiceProvider**: Represents an expert or service provider account within the **TuLink** platform, offering their skills and expertise.
- **StartUp**: Represents a startup profile within the **TuLink** platform, seeking expert consulting.
- **StartUpRepresentative**: Represents a representative of a startup within the **TuLink** platform.
- **InvoicePayment**: Represents payment information related to a service booking.
- **Booking**: Represents a scheduled consultation or booking made between a startup and a service provider.

5.3 Class Diagrams

The class diagrams presented in this section provide a visual overview of the data models and their attributes. This helps with understanding the overall schema of the project, and shows the relationships between models.

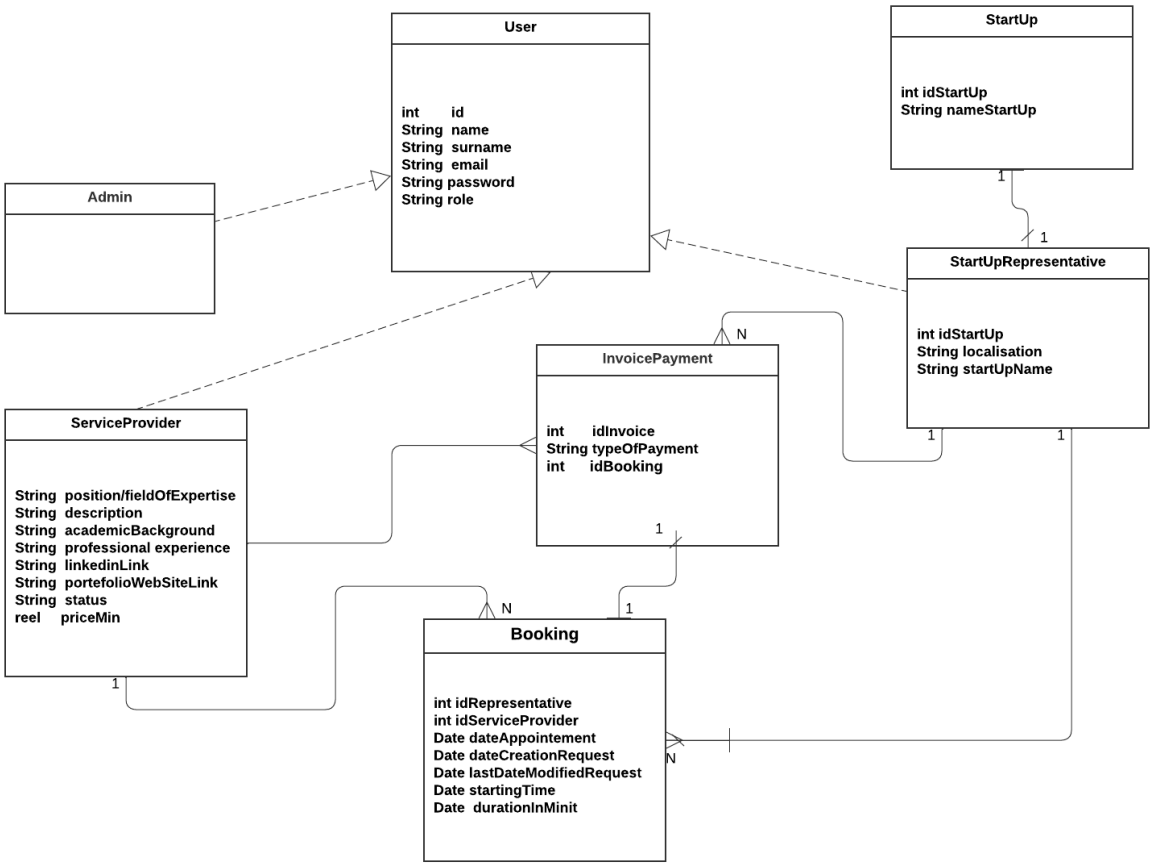

Figure 1: **Data Model Class Diagram**

6. API Endpoints

This section outlines the API endpoints that will be implemented in the **TuLink API** platform.

6.1 Service Provider Management

- **GET /serviceProvider/**: Retrieves all the service providers.
- **POST /serviceProvider/**: Create a new service provider profile.
- **GET /serviceProvider/{id}**: Retrieves specific details for the service provider with the given ID.
- **PUT /serviceProvider/{id}**: Updates profile information for a specific service provider.
- **DELETE /serviceProvider/{id}**: Delete a service provider profile using its ID.

```python
@serviceProvider.post("/serviceProvider/")
async def write_data(serviceProvider: ServiceProvider, db: Session = Depends(get_db)):
    try:
      print("allo",serviceProvider)
      password_hashed=bcrypt_context.hash(serviceProvider.password)
      result=db.execute(users.insert().values(
      name=serviceProvider.name,
      surname=serviceProvider.surname,
      email=serviceProvider.email,
      password=password_hashed,
      role="serviceProvider"
      ))
      print("good")
      db.execute(serviceProviders.insert().values(
      user_id=result.inserted_primary_key[0],
      positionFieldOfExpertise=serviceProvider.positionFieldOfExpertise,
      academicBackground=serviceProvider.academicBackground,
      description=serviceProvider.description,
      professionalExperience=serviceProvider.professionalExperience,
      linkedinLink=serviceProvider.linkedinLink,
      portefolioWebSiteLink=serviceProvider.portefolioWebSiteLink,
      status="waiting",
      priceMin=serviceProvider.priceMin
      ))
      db.commit()
      return {"message": "User created successfully"}
    except Exception as e:
        db.rollback()
        raise HTTPException(status_code=500, detail=f"Error inserting user: {e}")
```

Figure 2: **Service Provider Management Code Snippet**

6.2 Startup Management

- **GET /startup/**: Retrieves all startups.

- **POST /startup/**: Create a new startup profile.
- **GET /startup/{id}**: Retrieves specific details for the startup with the given ID.
- **PUT /startup/{id}**: Updates profile information for a specific startup.
- **DELETE /startup/{id}**: Delete a startup profile using its ID.

```python
@startUp.get("/startUp/",dependencies=[Depends(isAdmin)])
async def read_data(db: Session = Depends(get_db)):
    try:
        result = db.execute(
        startUps.select()
        ).fetchall()
        startUps_list = [{"id": row[0],
        "nameStartUp": row[1],
        "localisation": row[2]
    }
        for row in result
        ]
        return {"startUps": startUps_list}

    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Error retrieving startUps: {e}")
```

Figure 3: **Startup Management Code Snippet**

6.3 Startup Representative Management

- **GET /startupRepresentative/**: Retrieves all startup representatives.
- **POST /startupRepresentative/**: Create a new startup representative profile.
- **GET /startupRepresentative/{id}**: Retrieves specific details for the startup representative with the given ID.
- **PUT /startupRepresentative/{id}**: Updates profile information for a specific startup representative.
- **DELETE /startupRepresentative/{id}**: Delete a startup representative profile using its ID.

6.4 Booking Management

- **GET /booking/**: Retrieves all booking.
- **POST /booking/**: Create a new booking.
- **GET /booking/{id}**: Retrieves specific details for the booking with the given ID.
- **PUT /booking/{id}**: Updates profile information for a specific booking.

```python
@startUpRepresentative.put("/startUpRepresentative/{id}",dependencies=[Depends(isAdmin)])
async def modif_data(id:int,startUpRepresentative:StartUpRepresentative, db: Session = Depends(get_db)):
    try:
        user_db = db.execute(users.select().where(users.c.id == id)).fetchone()

        if user_db is None:
            raise HTTPException(status_code=404, detail="User not found")
        db.execute(users.update()
        .where(users.c.id == id)
        .values(
        name=startUpRepresentative.name,
        surname=startUpRepresentative.surname,
        email=startUpRepresentative.email,
        password=startUpRepresentative.password,
        role="startUpRepresentative"
        ))
        db.execute(startUpRepresentatives.update()
        .where(startUpRepresentatives.c.user_id == id).values(
        user_id=id,
        idStartUp=startUpRepresentative.idStartUp
      ))
        db.commit()
        updated_user = db.execute(users.select().where(users.c.id == id)).fetchone()
        return {"user": {"id": updated_user[0], "name": updated_user[1], "email": updated_user[2], "password": updated_user[3]}}

    except Exception as e:
        db.rollback()
        raise HTTPException(status_code=500, detail=f"Error updating user: {e}")
```

Figure 4: **Startup Representative Management Code Snippet**

- **DELETE /booking/{id}**: Delete a booking using its ID.

```python
@booking.get("/booking/{id}",dependencies=[Depends(isValid)])
async def read_data(id: int, db: Session = Depends(get_db)):
    try:
        result = db.execute(
        bookings.select()
        .where(bookings.c.id == id)
        ).fetchone()
        if result is None:
            raise HTTPException(status_code=404, detail="booking not found")

        print(result)
        booking ={
        "idRepresentative":result.idRepresentative,
        "idServiceProvider":result.idServiceProvider,
        "dateAppointement":result.dateAppointement,
        "dateCreationRequest":result.dateCreationRequest,
        "lastDateModifiedRequest":result.lastDateModifiedRequest,
        "startingTime":result.startingTime,
        "durationInMinit":result.durationInMinit
}

        return {"booking": booking}

    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Error retrieving bookings: {e}")
```

Figure 5: **Booking Management Code Snippet**

6.5 InvoicePayment Management

- **GET /invoicePayment/**: Retrieves all InvoicePayments.
- **POST /invoicePayment/**: Create a new InvoicePayment.
- **GET /invoicePayment/{id}**: Retrieves specific details for the InvoicePayment

with the given ID.

- **PUT /invoicePayment/{id}**: Updates profile information for a specific Invoice-Payment.
- **DELETE /invoicePayment/{id}**: Delete a InvoicePayment using its ID.

```python
@invoicePayment.delete("/invoicePayment/{id}",dependencies=[Depends(isValid)])
async def delete_data(id:int, db: Session = Depends(get_db)):
    try:
        invoicePayment_db = db.execute(invoicePayments.select().where(invoicePayments.c.id == id)).fetchone()
        if invoicePayment_db is None:
            raise HTTPException(status_code=404, detail="invoicePayment not found")
        db.execute(invoicePayments.delete().where(invoicePayments.c.id == id))
        db.commit()
        return {"message": f"invoicePayment with id {id} has been deleted successfully"}

    except Exception as e:
        db.rollback()
        raise HTTPException(status_code=500, detail=f"Error deleting invoicePayment: {e}")
```

Figure 6: **Invoice Payment Management Code Snippet**

7. SECURITY IMPLEMENTATION

This section details the security measures that have been implemented in the TuLink API Platform to protect user data and ensure secure access to its functionalities.

7.1 Password Hashing

To protect user passwords from potential breaches, the TuLink API implements a robust password hashing system:

- Passwords are hashed using the one-way bcrypt library.
- A unique salt is generated for each password before hashing.
- The generated hash, instead of the password, is stored in the database.

```
SECRET_KEY='197b2c37c391bed93fe80344fe73b806947a65e36206e05a1a23c2fa12702fe3
ALGORITHM='HS256'
bcrypt_context=CryptContext(schemes=['bcrypt'],deprecated='auto')
oauth2_bearer=OAuth2PasswordBearer(tokenUrl='auth/token')
```

Figure 7: **Password Hashing Code Snippet**

7.2 JWT (JSON Web Tokens) for Authentication

To enable secure authentication of API requests, the TuLink API uses JSON Web Tokens (JWTs):

- A JWT is generated after the user's credentials are verified.
- The token is included as a bearer token in the authorization header for subsequent requests.
- The JWT is cryptographically signed by the server, this ensures that the token is valid.
- JWTs have an expiration time, to avoid long-term misuse of stolen tokens.

```python
@router.post("/token",response_model=Token)
async def login_for_access_token(form_data:Annotated[OAuth2PasswordRequestForm,Depends()],
db:db_dependency):
    print(form_data.username,form_data.password)
    user=authenticate_user(form_data.username,form_data.password,db)
    if not user:
        raise HTTPException(status_code=status.HTTP_401_UNAUTHORIZED,detail='Could not validate user.')
    token=create_access_token(user.email,user.id,user.role,timedelta(minutes=90))
    return {'access_token':token,'token_type':'bearer'}

def authenticate_user(username:str,password:str,db):
    user = db.execute(users.select()
                      .where(users.c.email == username)).fetchone()
    if not user:
        print("if not user")
        return False
    if not bcrypt_context.verify(password,user.password):
        print("if not bcrypt_context.verify(password,user.password)")
        return False
    return user

def create_access_token(username:str,user_id:int,role:str,expires_delta:timedelta):
    encode={'sub':username,'id':user_id,'role':role}
    expires=datetime.utcnow()+expires_delta
    encode.update({'exp':expires})
    return jwt.encode(encode,SECRET_KEY,algorithm=ALGORITHM)
```

Figure 8: **JWT Generation Code Snippet**

8. CONCLUSION

The **TuLink** API, developed with a strong emphasis on security and user experience, successfully addresses the limitations within the Tunisian startup ecosystem's access to expert consultancy. By providing a unified platform for matchmaking, booking, and consultation management, this project establishes a solid foundation for facilitating efficient collaboration between Tunisian startups and local experts. The platform's implementation of bcrypt password hashing and JWT authentication demonstrates a strong commitment to security and maintaining user privacy.

This project has empowered startups by increasing their access to expertise, and it has provided experts with opportunities for growth. By bridging the gap between those who need guidance and those who have the expertise to offer, the **TuLink** API seeks to foster a more connected and user-centric ecosystem within Tunisia. This project serves as a demonstration of the potential for innovation to transform the way startups operate and connect in Tunisia.

9. PROSPECTS

The **TuLink** API platform lays the groundwork for future development and expansion, and it has the potential to be extended in many ways:

- **Enhanced Matching Algorithms:** Implementing machine learning algorithms can enhance matchmaking by providing more personalized and context-aware expert recommendations based on startup needs.
- **Integrated Communication Tools:** Adding features for direct communication between startups and experts, such as real-time chat and in-app video calls, would improve the user experience.
- **Advanced Payment Solutions:** Incorporate secure and integrated payment methods for streamlined transactions using STRIPE API which is an external API used for handling payments, subscriptions and payouts.
- **Expanded Data Analysis:** Implementing a more robust system for data analysis from both the startups and the experts can provide actionable insights and improve the platform's functionalities.

These future improvements will make **TuLink** a more complete and better suited for the specific needs of Tunisian startups and experts, while offering new ways to connect and collaborate.

REFERENCES

1. FastAPI Official Documentation. https://fastapi.tiangolo.com/
2. SQLAlchemy Official Documentation. https://www.sqlalchemy.org/
3. Python documentation. https://www.python.org
4. Postman Official Website. https://www.postman.com/
5. JWT Official Website. https://jwt.io/
6. Bcrypt library documentation. https://pypi.org/project/bcrypt/