**COM661: Full Stack Strategies and Development Semester Assignment – Application Plan**

**Project Title: Biz Directory**

**Student Name:**

**Student ID:**

**1. Application Overview**

**Name:** Biz Directory

**Purpose:**  
Biz Directory is a searchable directory of businesses and customer reviews. Users can view and search business listings and read reviews. Registered users can add their own businesses and reviews. Administrators can edit or delete any listing or review.

**Dataset:**  
The application uses data from the Yelp Dataset Challenge. The original dataset has over 160,000 businesses and 8 million reviews. We will use a smaller version with 60,000 businesses and all their reviews and tips.

**Dataset Source:**

<https://www.yelp.com/dataset>

**Note on Tips:** Tips from the dataset will be stored as part of the reviews collection. Each tip will be treated as a short review.

**2. Technologies Used**

A diagram of a system

AI-generated content may be incorrect.This project uses Python Flask for the backend, React for frontend and MongoDB for the database

.

|  |  |  |
| --- | --- | --- |
| **Layer** | **Technology** | **Purpose** |
| Backend | Python Flask | Creates the API and handles all requests |
| Database | MongoDB | Stores all data (users, businesses, reviews) |
| Database Connection | PyMongo / Flask-PyMongo | Connects Python code with MongoDB |
| Authentication | Flask-JWT-Extended | Manages user login with JWT tokens |
| Password Security | Werkzeug.security | Hashes passwords before saving |
| Testing | Postman | Tests all API endpoints |
| Configuration | python-dotenv | Manages secret keys and passwords |
| Frontend (Optional) | React.js or Postman | For testing and demonstration |

**3. Database Structure**

The database has three collections: Users, Businesses, and Reviews.

**Collection: Users**

{

"\_id": ObjectId,

"username": "john\_doe",

"email": "john@email.com",

"password": "hashed\_password",

"role": "user" or "admin",

"createdAt": Date

}

**Collection: Businesses**

{

"\_id": ObjectId,

"name": "Joe's Coffee",

"city": "New York",

"state": "NY",

"address": "123 Main St",

"category": "Coffee Shop",

"phone": "555-0123",

"rating": 4.5,

"reviewCount": 25,

"createdAt": Date

}

**Collection: Reviews**

{

"\_id": ObjectId,

"businessId": ObjectId,

"userId": ObjectId,

"rating": 4,

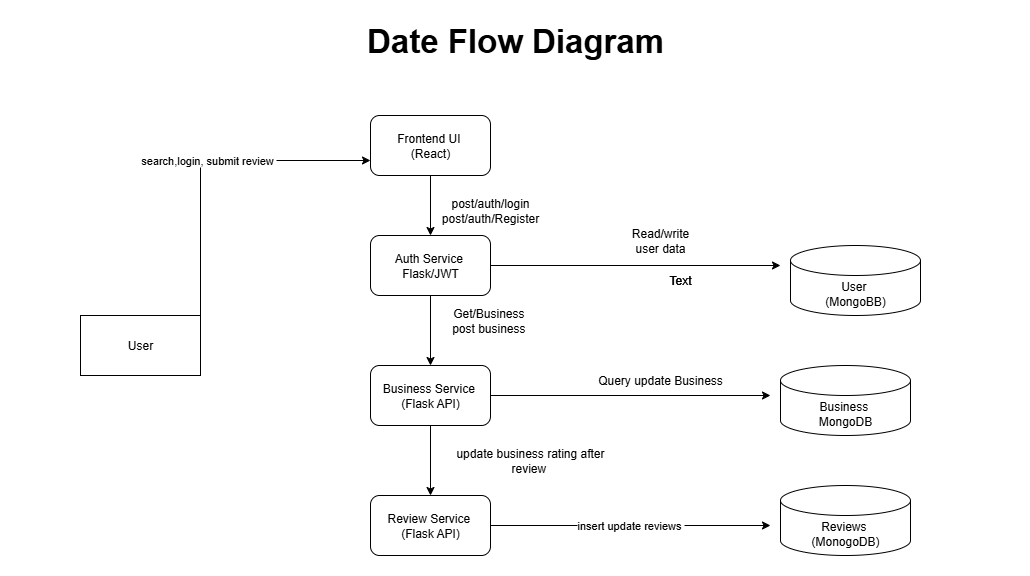
"text": "Great coffee and service!",

"createdAt": Date

}

**Note:** The business rating is automatically calculated from all its reviews. When a new review is added, the business rating updates.

**4. API Functionality**

****

**Authentication Endpoints**

|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Description** |
| POST | /api/auth/register | Create new user account |
| POST | /api/auth/login | Login and get JWT token |
| GET | /api/auth/me | Get current user information |

**Business Endpoints**

|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Description** |
| GET | /api/businesses | Get all businesses (with pages) |
| GET | /api/businesses/search | Search by name, city, or category |
| GET | /api/businesses/<id> | Get one business by ID |
| POST | /api/businesses | Add new business (login required) |
| PUT | /api/businesses/<id> | Edit business (admin only) |
| DELETE | /api/businesses/<id> | Delete business (admin only) |

**Review Endpoints**

|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Description** |
| GET | /api/businesses/<id>/reviews | Get all reviews for a business |
| POST | /api/businesses/<id>/reviews | Add new review (login required) |
| PUT | /api/reviews/<id> | Edit own review (login required) |
| DELETE | /api/reviews/<id> | Delete review (admin only) |

**Search Features:**

Users can search businesses by name, city, state, or category. Results can be filtered by rating.

**5. User Roles**

A diagram of a company

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| **Role** | **What They Can Do** |
| Public (No login) | View and search businesses and reviews |
| Registered User | Add businesses, add and edit their own reviews |
| Administrator | Edit or delete any business or review |

**6. Security Features**

* **JWT Authentication:** Users get a token after login. This token is needed for protected actions.
* **Password Hashing:** All passwords are encrypted before saving to database.
* **Role-Based Access:** Different permissions for regular users and admins.
* **Input Validation:** Checks all data before saving to prevent errors.
* **Environment Variables:** Keeps secret keys safe in .env file.

**7. Implementation Timeline**

|  |  |  |
| --- | --- | --- |
| **Week** | **Task** | **Details** |
| Week 1 | Setup | Install Flask, connect MongoDB, create project folders |
| Week 2 | User Management | Build register, login, and JWT system |
| Week 3 | Business Endpoints | Create all business CRUD operations |
| Week 4 | Review Endpoints | Create all review CRUD operations |
| Week 5 | Data Import | Load Yelp data into MongoDB |
| Week 6 | Testing | Test all endpoints using Postman |
| Week 7 | Documentation | Write README and export Postman docs |
| Week 8 | Final Submission | Record demo video and submit files |

**8. Testing and Documentation**

* **Postman Testing:** All endpoints will be tested with Postman test collection.
* **Postman Documentation:** API documentation will be generated and exported as PDF.
* **Error Handling:** Clear error messages with proper HTTP status codes (400, 401, 404, 500, etc.).
* **README File:** Step-by-step instructions for running the application.