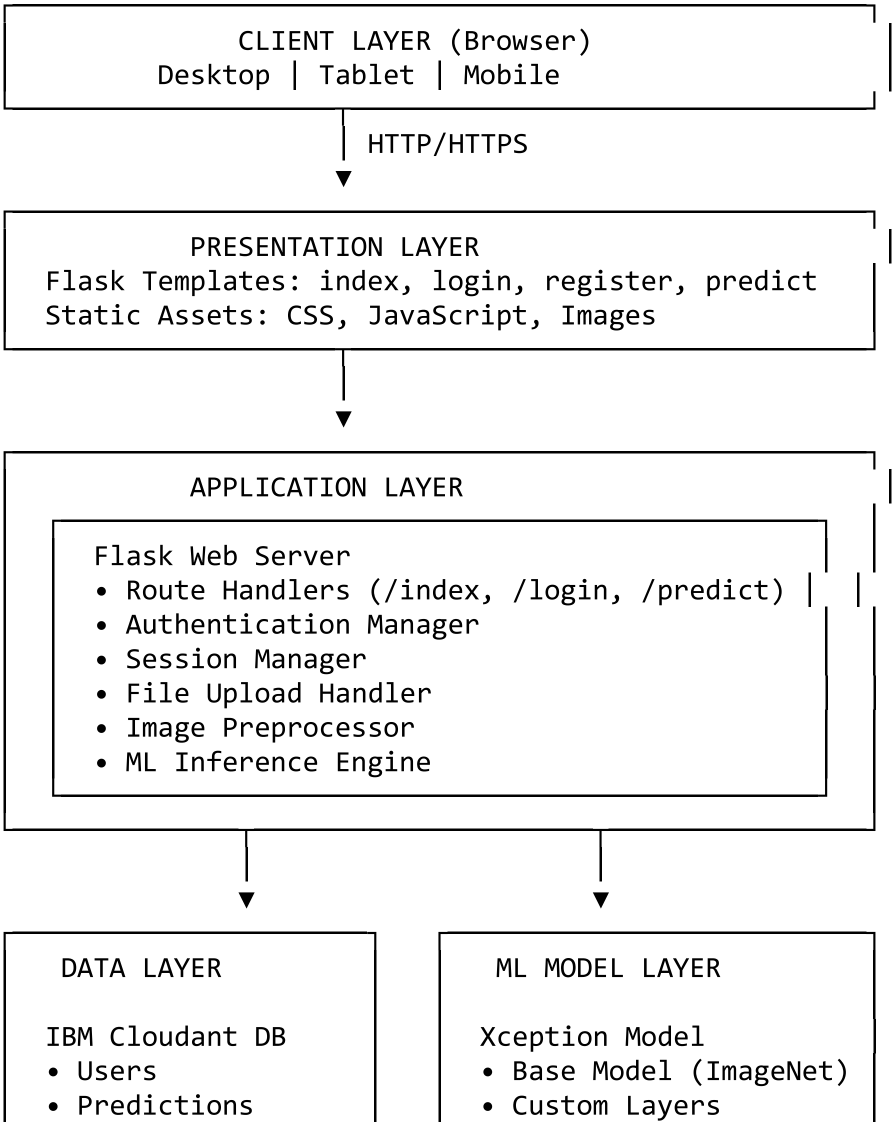
Solution Architecture

Diabetic Retinopathy Detection System

ARCHITECTURE OVERVIEW

Type: Three-Tier Web Application Architecture - **Presentation Tier:** Web-based user interface - **Application Tier:** Flask backend with ML model - **Data Tier:** IBM Cloudant database and file storage

HIGH-LEVEL ARCHITECTURE



CLIENT LAYER (Browser)
Desktop | Tablet | Mobile

HTTP/HTTPS

PRESENTATION LAYER
Flask Templates: index, login, register, predict
Static Assets: CSS, JavaScript, Images

APPLICATION LAYER

Flask Web Server
- Route Handlers (/index, /login, /predict)
- Authentication Manager
- Session Manager
- File Upload Handler
- Image Preprocessor
- ML Inference Engine

DATA LAYER

IBM Cloudant DB
- Users
- Predictions

ML MODEL LAYER

Xception Model
- Base Model (ImageNet)
- Custom Layers

```
│                       │    │ • Input: 299x299x3          │
│ File Storage          │    │ • Output: 5 classes         │
│ • /uploads            │    │ • Size: 88 MB               │
│ • /model              │    │                             │
│ • /static             │    │                             │
└───────────────────────┘    └─────────────────────────────┘
```

COMPONENT ARCHITECTURE

Frontend Components

1. **Navigation Bar** - Logo, login/logout buttons, responsive menu
2. **Home Page** - Welcome section, system overview, dynamic content
3. **Authentication Pages** - Login/registration forms, logout confirmation
4. **Prediction Interface** - File upload, image preview, results display
5. **Static Assets** - Custom CSS, JavaScript, images

Backend Components (app.py)

1. **Route Handlers** - Handle HTTP requests for all pages
2. **Authentication Module** - User registration, login, session management
3. **File Management** - Upload handling, validation, secure storage
4. **Image Processing** - Loading, resizing (299x299), normalization
5. **ML Inference** - Model loading, prediction, result interpretation
6. **Database Interface** - Cloudant connection, CRUD operations

ML Components (train_model.py)

1. **Base Model** - Xception pre-trained on ImageNet
2. **Custom Head** - GlobalAvgPool → Dense(1024) → Dense(512) → Dense(256) → Dense(5)
3. **Training Pipeline** - Data augmentation, batch processing, callbacks
4. **Inference Pipeline** - Model loading, preprocessing, forward pass

DATA FLOW

User Registration Flow

```
User → Form → Validate → Check Uniqueness → Store in DB → Success
```

User Login Flow

```
User → Credentials → Query DB → Validate → Create Session → Redirect
```

Prediction Flow

```
User → Upload Image → Validate File → Save → Preprocess →
Load Model → Inference → Extract Results → Store in DB → Display
```

SECURITY ARCHITECTURE

Authentication & Authorization

- Session-based authentication with Flask sessions
- Secure session cookies with encryption
- Protected routes requiring login
- Session validation on each request

Data Security

- File type and size validation
- Secure filename handling (Werkzeug)
- IAM authentication for Cloudant
- Input sanitization

DEPLOYMENT ARCHITECTURE

Development Environment

```
Local Machine → Python 3.8+ → Flask Dev Server (port 5000) →
Local Storage → IBM Cloudant (Cloud)
```

Production Options

1. **IBM Cloud** - Cloud Foundry, integrated Cloudant, auto-scaling
2. **Docker** - Containerized deployment, portable
3. **Traditional Server** - Nginx + Gunicorn + Flask

SCALABILITY

Horizontal Scaling: - Multiple Flask instances behind load balancer - Stateless application design - Shared cloud storage

Vertical Scaling: - Increase server resources (CPU, RAM) - GPU acceleration for inference

---

TECHNOLOGY MAPPING

Layer	Technology	Purpose
Frontend	HTML5, CSS3, Bootstrap, JS	User interface
Backend	Python, Flask	Web framework
ML	TensorFlow, Keras, Xception	Deep learning
Database	IBM Cloudant	NoSQL storage
Storage	Local File System	Image files
Security	Werkzeug, Flask Sessions	Authentication

DESIGN PATTERNS

- **MVC Pattern** - Model-View-Controller separation
- **Layered Architecture** - Clear separation of concerns
- **Singleton** - Model loading (single instance)
- **Decorator** - Flask route decorators

PERFORMANCE OPTIMIZATION

Frontend: Minified CSS/JS, image optimization, lazy loading
Backend: Lazy model loading, efficient preprocessing, query optimization
ML: GPU acceleration, efficient model architecture

KEY ARCHITECTURAL DECISIONS

1. **Flask over Django** - Lightweight, flexible, easier ML integration
2. **Xception over ResNet** - Better accuracy-speed tradeoff
3. **Cloudant over SQL** - Scalability, JSON documents, managed service
4. **Session-based auth** - Simpler implementation for web app
5. **Lazy model loading** - Faster startup time
6. **Web app over mobile** - Broader accessibility, easier deployment