Project Cloud Computing

1 Outline

You are a cloud computing marvel! Your architecture and planning has left the CEO of the **LowTech GmbH** at awe! Nice job! Since you are the responsible **cloud consultant, architect, engineer** and **cloud god** you now have to prepare a tech demo for the CEO!

You are asked to prepare a **practical implementation** of a showcase for the **Webshop** in a CSP plattform!

The requirements of your boss are the following:

- A Web Frontend for the shop with basic functionality (see section 5)!
- A Middleware with REST-API
- A Storage Backend (relational database or non-relational database)
- A Storage Backend for BLOB (Binary Large Objects)!
- High Availability with Load Balancing!

The demo should demonstrate a **Three-Tier application architecture** and should use functionality of the CSP (Cloud Service Provider)! Use appropriate services of the CSP and implement them in the demonstration for the CEO of the LowTech GmbH!

2 Implementation details for the Webshop

- Use a modern and suitable technology for the dynamic web front end!
- Use Python 3 or Java as a programming language for the middleware!
- Use a modern and suitable technology for the REST endpoint in the middleware!
- Separate the application into different tiers (Presentation-Tier, Application-Tier, Data-Tier)!
- Use suitable runtime platforms of the CSP for your applications (e.g. virtual machines or containers)!
- Use a service offering of your choice for the storage backend of your application! (*It does not matter if it is relational or NoSQL!*)
- Use a load balancer for the high availability of the application! (Think of designs for your application, that make it highly available)!
- **You do not need to implement a login mechanism or security measures, since this is a demo application!** (more in section 5)

3 Deliverables

Artifacts for third Milestone:

1. Detailed description of your application with software design and design decisions!
2. Diagrams of your demo application (Context and Scope, Building Block View, Runtime View, Deployment View[1])!
3. Detailed description of the development and installation process in the public Cloud infrastructure of your choice (see section 4)!
4. Critical analysis of your demo application in relation to the use of cloud services and a detailed demonstration of the application in the presentation and report!
5. Mandatory use of GitHub for the development of your application. The repository should be publicly visible and a continuous commit history should be visible!

Prepare a demonstration of your demo application for the final presentation and show the use of appropriate public CSP services in your demo. The use of GitHub for the development is **mandatory** and the individual contribution of **each team member** has an impact on the evaluation and the final grade! The **individual contributions** of each team member will be evaluated individually after submission by inspecting the commits to the repository!

4 Choosing a CSP

Choose a public cloud service provider according to the following formula:

$$a \equiv b \mod 3 \tag{1}$$

With `b` being your **group number** and `a` as the result for your CSP:

- `a = 0` → **Amazon Web Services**
- `a = 1` → **Google Cloud Platform**
- `a = 2` → **Microsoft Azure**

Prepare a final report of 25-30 pages and a presentation of 30 minutes length on the entire project! Include the findings of every milestone in your report and presentation! You must include a demo of your implementation in the presentation! The report, presentation and demo are due to **28.02.2025**!

[1] See arc42 template specification: `https://arc42.org/overview`

5 Architecture details for Webshop Demo

A Three-Tier architecture divides the application into three distinct layers: the Presentation-Tier, the Application-Tier, and the Data-Tier [1]. This separation enhances maintainability, scalability, and security.

Requirements and functionalities for the **Three-Tier** demo web application **Webshop**:

1. **Presentation-Tier (Frontend)**
   User Interface (UI):

   - **Product Catalog:** Display products in various categories with images, names, descriptions, and prices. (see block description on page 4)
   - **Product Search and Filter:** Provide (simple) search functionality and filter options (e.g. by price, category).
   - **Product Details:** Detailed view of a selected product with comprehensive information.
   - **Shopping Cart:** Ability to add products to the cart, change quantities, and remove items.
   - **Checkout Process:** Step-by-step checkout process (entering address, selecting payment methods, order review).
   - **Responsive Design:** Optimize the user interface for various devices (desktop, tablet, mobile).

   Technologies:
   HTML, CSS, JavaScript Frameworks like **React**, **Angular**, or **Vue.js** for dynamic and interactive UI components.

2. **Application-Tier (Backend)**
   Business Logic:

   - **Product Management:** CRUD operations (Create, Read, Update, Delete) for products.
   - **Order Management:** Process orders, track order status, and manage orders.
   - **Payment Processing:** Integrate with payment gateways (e.g., PayPal, Stripe) for transaction handling **(Just a Mock-up!!!)**.
   - **Inventory Management:** Manage stock levels and notify when stock is low.
   - **Email Notifications:** Automatically send order confirmations and shipping notifications.

Technologies:
Programming language Python 3 (e.g., **Django** or **Flask**) or Java (**Spring Boot**). RESTful APIs for communication with the frontend.

3. **Data-Tier (Database)**
   Databases:

   - **Product Data:** Tables for product information, categories, prices, etc.
   - **Order Data:** Tables for order information, order status, payment methods, etc. Tables for order information, order status, payment methods, etc.
   - **Inventory Data:** Tables for stock levels and supplier information.

   Technologies:
   ORM (Object-Relational Mapping) like **SQLAlchemy** or **Hibernate**.
   Relational databases like **MySQL**, **PostgreSQL**, or **MariaDB**.
   Or NoSQL databases like **MongoDB**.
   Storage Backend for unstructured data like BLOB.

Additional Requirements:

- **Performance and Scalability:** Load balancing and horizontal scalability options.

Product catalog and contents of Webshop application

The **product catalog** does not need to be very detailed! However it should be sufficient to have a demonstration of the basic functionalities of the application. **10** items for the product catalog should be sufficient!

Example Architecture:

1. **Presentation-Tier (Frontend):**
   A React application implemented in an GCP instance type `n2-standard-4`.
2. **Application-Tier (Backend):**
   A Python 3 application server using Django managing business functions and providing API endpoints implemented in the Google App Engine.
3. **Data-Tier (Database):**
   A Cloud SQL database storing product and order data running in GCP. BLOB is stored in Google Cloud Storage.

References

[1] M. Richards and N. Ford, Fundamentals of Software Architecture: An Engineering Approach. O'Reilly Media, Incorporated, 2020.