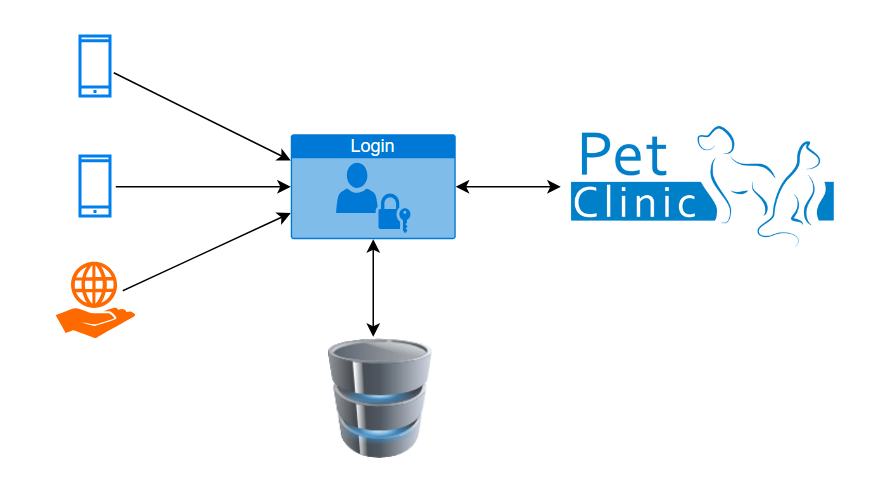
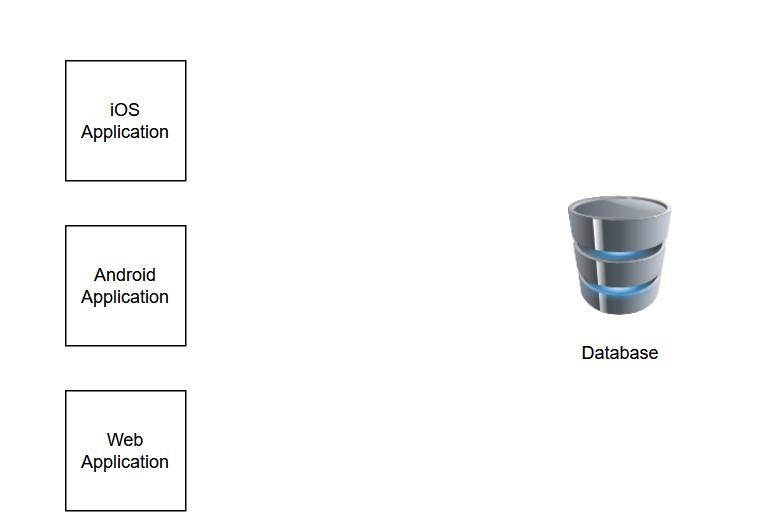
Very High-level diagram



3 Tier architecture for Spring pet clinic

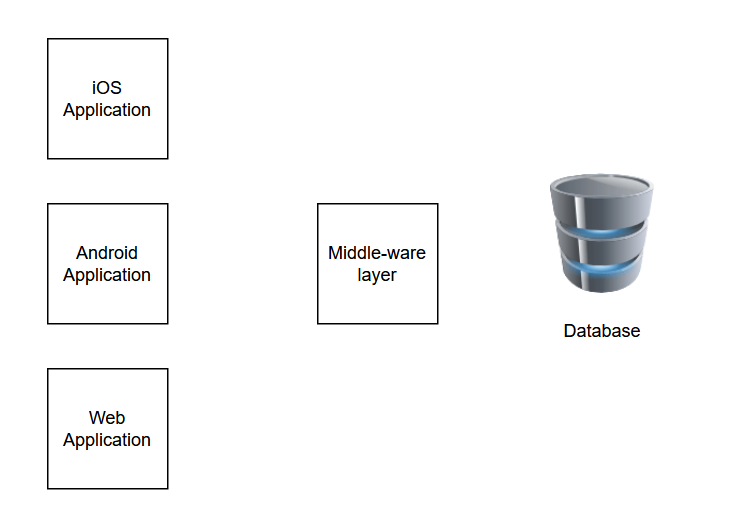


3) Now, Consideration on application

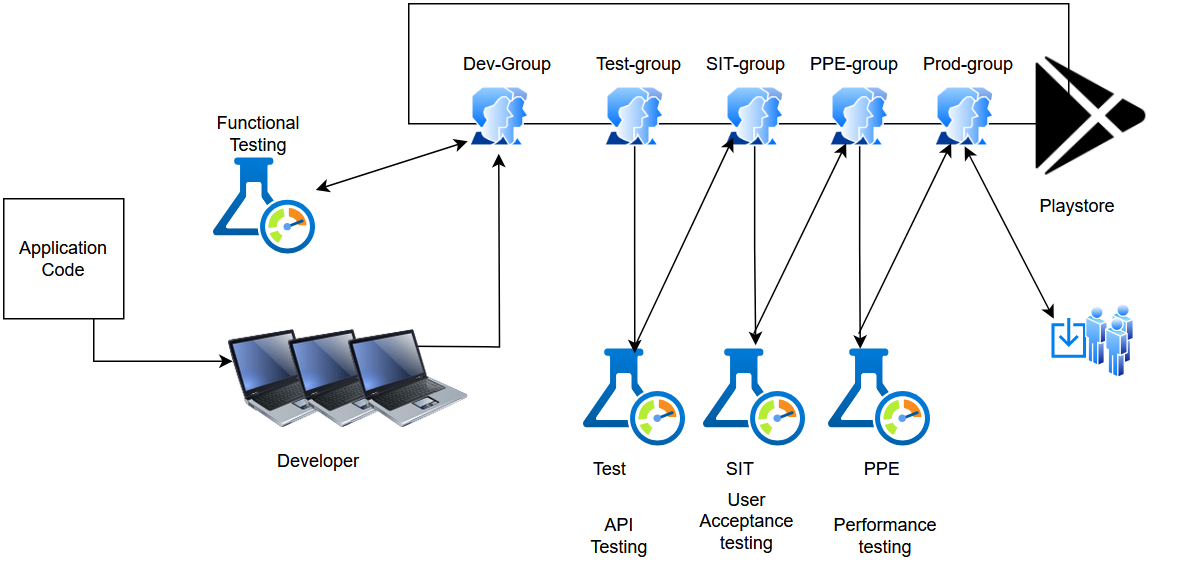
1) My application is available over internet which may get impacted by hacker, so I need to secure my application.

2) Only authorized user allows to do needful things.

3) We need to manage autoscaling, and few other infrastructure components. All these things would be taken care at middleware layer.



4) My application is having 5 Environment in frontend, middleware and database. Help me understand the flow.



**1) TestFlight for iOS/Android Application**

TestFlight is Apple’s beta testing platform for iOS, iPadOS, macOS, watchOS, and tvOS apps. It allows developers to distribute pre-release versions of their applications to testers and gather feedback before the official release.

* **Key Features:**
  + Supports up to **10,000 external testers** via email invites or public links.
  + Allows developers to push multiple builds and collect user feedback.
  + Integrated with **App Store Connect**.
  + Supports automatic crash reporting.
* **For Android:** The equivalent is **Google Play Beta Testing**, managed via Google Play Console.
* **Relevance to DevOps:**
  + Useful for **CI/CD pipelines** where new builds can be automatically uploaded to TestFlight for testing before release.
  + Integrates well with **Azure DevOps or Jenkins for mobile application deployments**.

**2) Functional Testing**

Functional testing ensures that an application works according to its specifications and meets business requirements.

* **Types of Functional Testing:**
  1. **Unit Testing:** Tests individual components (e.g., functions, classes).
  2. **Integration Testing:** Checks if multiple modules work together.
  3. **System Testing:** Evaluates the application as a whole.
  4. **Regression Testing:** Ensures new changes don’t break existing functionality.
* **Example:**
  1. In your **iOS app with Azure Function backend**, functional testing would check if the API endpoints correctly return data and the mobile app processes it as expected.
* **Tools:** Selenium, Appium (for mobile), Postman, JUnit, TestNG.

**3) API Testing**

API testing verifies that application programming interfaces (APIs) work correctly, return expected data, and handle edge cases.

* **Key Aspects:**
  + Validating **response status codes** (200 OK, 400 Bad Request, etc.).
  + Checking **data accuracy** in responses (JSON/XML formats).
  + Verifying **authentication and authorization** (OAuth, JWT, etc.).
  + Handling **rate limiting** and **error scenarios**.
* **Example:**
  + Your **Azure Function App backend** exposes APIs through **Azure API Management (APIM)**. API tests would ensure that endpoints return expected responses under different conditions.
* **Tools:** Postman, RestAssured, SoapUI, JMeter (for API load testing).

**4) Performance Testing**

Performance testing evaluates how well an application performs under different conditions, such as high user load, network fluctuations, or limited computing resources.

* **Types of Performance Testing:**
  1. **Load Testing:** Measures app behavior under expected load.
  2. **Stress Testing:** Tests system behavior under extreme conditions.
  3. **Spike Testing:** Observes system behavior during sudden traffic spikes.
  4. **Scalability Testing:** Measures system’s ability to scale.
  5. **Soak Testing:** Evaluates system performance over extended periods.
* **Example:**
  1. When you run **JMeter tests in Azure DevOps**, you analyze **CPU usage metrics in Azure Application Insights** to correlate performance with system autoscaling.
* **Tools:** JMeter, Locust, k6, Gatling, Azure Load Testing (ALT).

**5) User Acceptance Testing (UAT)**

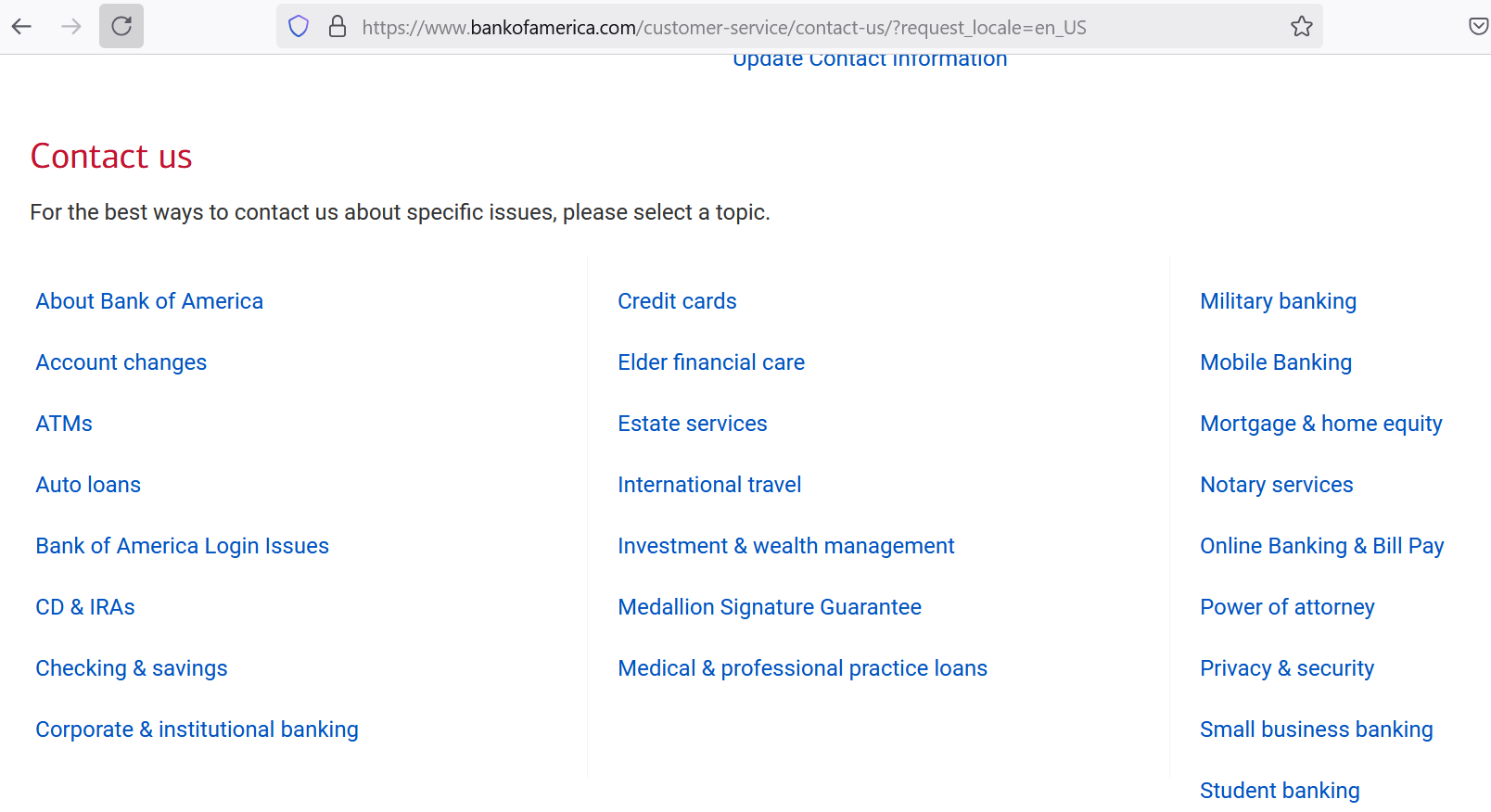
UAT is the final phase of testing where **real end-users** validate if the system meets business requirements before production release.

* **Characteristics:**
  + Conducted by **business stakeholders or clients**.
  + Focuses on **real-world use cases** rather than technical bugs.
  + Determines if the application is **ready for deployment**.
* **Example:**
  + If you build a payment processing system, the finance team would perform UAT by simulating real transactions to ensure the system works as expected.
* **Tools:** TestRail, Azure Test Plans, Jira.

**6) End-User and Its Support Model**

An **end-user** is the person who directly interacts with your software or service.

* **End-User Support Models:**
  1. **Self-Service Support:** FAQs, knowledge bases, and chatbots (e.g., Azure AI Chatbot).
  2. **Helpdesk Support:** Ticketing system (e.g., ServiceNow, Jira Service Desk).
  3. **Live Support:** Email, phone, or chat-based assistance.
  4. **On-Site Support:** IT teams providing direct support in an enterprise setup.
* **Example in DevOps:**
  1. If an issue is reported in production, you might check **Azure Application Insights logs** to diagnose the problem before escalating to L2 or L3 support teams.



Point to be noted: Environment/tools might might not be unform.