Technische Universität Braunschweig

ISF

Verifiable Paper Certificates

Android- und Serverapplikation

Software-Entwicklungspraktikum (SEP)
Sommersemester 2015

Testspezifikation

Auftraggeber
Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund
Prof. Dr.-Ing. Lars Wolf
Mühlenpfordtstraße 23
38106 Braunschweig

Betreuer: Björn Gernert, Dominik Schürmann

Auftragnehmer:

Name	E-Mail-Adresse
Osama Alrjoob	y0067649@tu-bs.de
Tim Brandes	y0067685@tu-bs.de
Max-Frederik Eckardt	y0058622@tu-bs.de
Linda Fliss	y0067315@tu-bs.de
Thomas Haas	y0068180@tu-bs.de
Jan-Frederick Musiol	y0067262@tu-bs.de
Jan Schlichter	y0067112@tu-bs.de
Tim Schubert	y0067212@tu-bs.de

Braunschweig, 15. Juli 2015

Inhaltsverzeichnis

1 Einleitung

Das vorliegende Dokument enthält die vollständige Testspezifikation des Softwaresystems "Cert-Check" im Rahmen des Softwareentwicklungspraktikum 2015. Das Softwaresystem besteht aus zwei Komponenten, die in unterschiedlichen Umgebungen laufen. Einer Client-Komponente und einer Server-Komponente.
Die Client-Komponente ist eine Android Applikation die auf einem Smartphone mit Android 5.0 oder höher läuft. Sie liest ein vorliegendes Zeugnis mittels OCR ein, berechnet daraus einen Hash-Wert und schickt diese Daten, mit einer eindeutigen Identifikationsnummer der SecureID, über eine über SSL verschlüsselte Verbindung an die Server-Komponente.
Die Server-Komponente besteht aus einer Django-Webanwendung und einer Datenbank die zusammen auf einem zentralen Linux Server laufen. Die Django-Webanwendung nimmt Anfragen der Android Applikation entgegen und vergleicht sie mit den Daten in der Datenbank. Außerdem werden die Daten in der Datenbank über ein Webinterface, welches die Django-Webanwendung zur Verfügung stellt, verwaltet. Um diese Aufgaben übernehmen zu können muss sowohl die Server-Komponente als auch die Client-Komponente strengen Qualitätsstandards, insbesondere in Hinsicht auf Richtigkeit der Funktion, Zuverlässigkeit des Ergebnisses und Grad der Sicherheit, genügen. Um die Erreichung dieser Ziele sicherstellen zu können ist es wichtig, dass die Software über eine hohe Analysierbarkeit und Überprüfbarkeit verfügt.
Die Testspezifikationen definieren Abnahmetests, Integrationstest und Unit-Tests. Im Testplan werden die zu testende Komponenten und Funktionen identifiziert und adäquate Testfälle festgelegt. Des weiteren wird eine allgemeine Vorgehensweise für die einzelnen Testobjekte beschrieben. Der Aufbau der Testspezifkation entspricht dem IEEE Standard 829 für Software-Testdokumentation.

2 Testplan

Im Folgenden wird der Testplan des Softwaresystems beschrieben. Hierbei soll auf die einzelnen Funktionen eingegangen werden, die die Software bieten soll und welche Qualitätsanforderungen und funktionale Anforderungen auf welche Weise jeweils sichergestellt werden sollen.

Als allgemeine Risiken bei der Durchführung des Projekts seien Krankheit der beteiligten Personen, sowie unerwartete Fehlfunktionen verwendeter Drittsoftwarebibliotheken genannt.

In erster Instanz sei immer die Person für die Überprüfung der Anforderungen einer Komponente verantwortlich, welche die zu testenden Änderungen an ihr vorgenommen hat.

2.1 Zu testende Komponenten

In diesem Dokument werden folgende Komponenten getestet und analysiert:

Android Applikation:

- **Image**
  Lässt den Nutzer ein Bild von einem Zeugnis aufnehmen oder aus dem Telefonspeicher auswählen.
- **ImageProcessing**
  Wandelt das erhaltene Bild eines Zeugnisses in den darauf enthaltenen Text um.
- **Data**
  Speichert den erkannten Text temporär nach Kontext sortiert ab. Enthält außerdem die globalen Einstellungen der App.
- **Verifikation**
  Lässt den Nutzer die erkannten Daten aus dem Zeugnis überprüfen und gleicht diese dann mithilfe der Communication Komponente mit der Datenbank des Servers ab. Anschließend zeigt sie das Ergebnis an.
- **Communication**
  Verwaltet auf der Client-Seite die Kommunikation mit dem Server.

Server:

- **Authentifikation**
  Bietet die Möglichkeit sich als autorisierter Nutzer einzuloggen.
- **Views**
  Bietet eine grafische Oberfläche für autorisierte Nutzer, um Zeugnisse und andere Daten in der Datenbank anzulegen, zu löschen oder zu modifizieren.Weiterhin können über diese Schnittstelle auch Zeugnisse als PDF generiert werden.
- **Verification**
  Ermöglicht das Verifizieren von Zeugnissen, auch für nicht autorisierte Nutzer.
- **Models**
  Verwaltet die Zugriffe auf die Datenbank und bietet in Form von Models eine vereinfachte Schnittstelle um auf den Daten der Datenbank zu arbeiten.
- **Rest-API**
  Stellt eine REST-API bereit mit der die Daten geändert werden können.

2.2 Zu testende Funktionen/Merkmale

Folgende Funktionen/Merkmale sind durch die Testverfahren zu testen.

- **Foto machen F10**
- **Foto aus der Galerie auswählen F20**
- **Text erkennen F30**
- **Text anzeigen F40**
- **Hash generieren F50**
- **Daten zum Server senden F60**
- **Daten von dem Server empfangen F70**
- **Einloggen F80**
- **Zeugnis erstellen F90**
- **PDF downloaden F100**
- **Hash validieren F110**
- **Zeugnis verwalten RM8**
- **Zeugnis ändern RS8**

2.3 Nicht zu testende Funktionen

Wir werden die zugrunde liegende Software, die zum Betrieb und Erstellen unserer Software dient, nicht Testen. Dazu gehört z.B. das Betriebssystem auf der unsere Software läuft und die Tools mit der wir unsere Software geschrieben und kompiliert haben. Außerdem werden wir Softwarebibliotheken von Dritten nicht Testen, ihre Korrektheit wird Vorausgesetzt.

- **Google Android**
- **Google Android SDK**
- **Google Android Studio**
- **Tess Two**
- **Leptonica**
- **Django Framework**
- **Texlive**
- **Apache Server**
- **Ubuntu Linux**
- **SQLite**

2.4 Vorgehen

Im folgenden wird das Vorgehen für die einzelnen Testverfahren beschrieben, dabei wird das Vorgehen und die Art der Durchführung der Tests detailliert beschrieben.

2.4.1 Komponententests/Unit Tests

Die GUI lastigen Komponenten der Android Applikation und der Server Applikation werden mittels Testfällen getestet. Menschliche Tester testen die Komponenten, wie es in den Testfällen beschrieben ist, und protokollieren ihre Ergebnisse schriftlich. Kritische Funktionalitäten der Android Applikation und der Server Applikation werden von Testskripten getestet. So ist nach erfolgreichem Abschluss der Tests die unabhängige Funktionalität einer Komponente sichergestellt.

2.4.2 Integrationstest

Nach erfolgreichen Komponententests folgt die Integration der einzelnen Komponenten mittels Bottom-Up-Prinzip. Nach dem zusammenführen einzelner Komponenten wird die Funktionalität von Menschlichen Testern überprüft, die nach festgelegten Vorgehensweisen die Applikation testen. Ihre Beobachtungen werden schriftlich Protokolliert.

2.4.3 Abnahme und Funktionstests

Die Testfälle des Abnahmetests sollen gemeinsam mit dem Auftraggeber durchgeführt werden. Hierzu erhält der Auftraggeber vom Auftragnehmer vor Beginn der Abnahmetests eine Testanleitung, die beschreibt wie die im Pflichtenheft festgelegten Funktionen auf auf ihre Umsetzung hin geprüft werden können. Hierbei sollen insbesondere alle beschriebenen Anwendungsfälle wenigstens einmal ausgeführt werden. Hierzu zählen vor allem aber nicht ausschließlich das Eintragen der Zeugnisse in die Zeugnisverwaltung und die Überprüfung des Zeugnisses mithilfe der Client-Software.

2.5 Testumgebung

Für die Komponententests und Integrationstests werden auf Seiten der App JUnit und die Android Test Suites genutzt. Für die Komponententests des Servers wird das Python-Modul `unittest` genutzt und zudem die Testbibliotheken, die in `djangorestframework` und `django` enthalten sind. Um Integrationstests durchführen zu können wird eine Serversoftware mit einer Netzwerkverbindung zur Client-Komponente benötigt. Zur Auslieferung einer Webseite zur Eingabe von Zeugnisdaten wird ein Webserver benötigt. Die Client-Software muss auf einem Gerät mit Android-Software laufen. Die Serversoftware benötigt die im Technischen Entwurf ausführlich beschriebenen Voraussetzungen. Zum Abnahmetest der Verifikationsschnittstelle, der Zeugnisverwaltung und der REST-API eignet sich auch ein handelsüblicher Web-Browser.

3 Abnahmetest

Der Abnahmetest soll die Erfüllung der im Pflichtenheft festgelegten Anforderungen auf Vollständigkeit und Erfüllung überprüfen. Hierdurch soll sichergestellt werden, dass das Produkt den Anforderungen des Auftraggebers entspricht und in dessen Umgebung eingesetzt werden kann. Ziel dieser Tests ist also die erfolgreiche Abnahme des Produktes durch den Kunden, nachdem dieser sich von der Qualität der Software durch die Abnahmetests überzeugen konnte.

3.1 Zu testende Anforderungen

In der folgenden Tabelle werden die zu testenden Anforderungen aufgezählt und ihre Testfälle gennant.

Nr	**Anforderung**	**Testfälle**
1	<F10> Foto aufnehmen	<T19>
2	<F20> Foto aus der Galerie auswählen	<T28>
3	<F30> Text erkennen	<T20>, <T23>
4	<F40> Text anzeigen	<T21>
5	<F50> Hash generieren	<T04>, <T05>
6	<F60> Daten zum Server senden	<T01>, <T24>, <T18>
7	<F70> Daten von Server empfangen	<T01>, <T24>, <T18>
8	<F80> Einloggen	<T01>
9	<F90> Zeugnis erstellen	<T09>,
10	<F100> PDF downloaden	<T10>
11	<F110> Hash validieren	<T25>, <T18>, <T15>, <T14>, <T11>
12	<RM8> Zeugnis verwalten	<T02>, <T03>, <T06>,
13	<RS5> Zeugnis ändern	<T03>, <T17>
14	<RS1> Unterstützung verschiedener Android-Geräte	<T27>, <T17>

3.2 Testverfahren

Die Testfälle des Abnahmetests sollen gemeinsam mit dem Auftraggeber durchgeführt werden. Hierzu erhält der Auftraggeber vom Auftragnehmer vor Beginn der Abnahmetests eine Testanleitung, die beschreibt wie die im Pflichtenheft festgelegten Funktionen auf auf ihre Umsetzung hin geprüft werden können.

Für nicht durch Testframework zu testende Testfälle gilt zudem: Die Eingabe Zeugnisse wird durch beispielhafte Zeugnisdaten getestet und die Eingabe in der Datenbank geprüft. Die erzeugte PDF-Datei muss manuell auf Richtigkeit geprüft werden. Die Authentifizierung wird durch Beispielaccounts und unterschiedliche Anmeldeversuche geprüft. Die Schnittstelle zur Validierung von SecureID und Hash-Wert wird durch Beispieldaten geprüft, die zufällig generiert werden können, oder geraten werden und das Ergebnis mit Wissen um den tatsächlichen Inhalt der Serverdatenbank überprüft. Das Nutzerinterface der Client-Software wird manuell bedient, um einen Benutzer der Software zu simulieren und so auf Bedienbarkeit zu testen. Die Benutzbarkeit und Verständlichkeit soll durch zumindest 5 Testpersonen, die nicht an der Entwicklung beteiligt sind, Stichprobenweise feststellt werden.

Zunächst müssen testweise Beispielzeugnisse über die Eintrageschnittstelle eingetragen worden sein, anschließend kann die Verifikationskomponente mit der Client-Anwendug geprüft werden.

Zunächst wird die Funktion der Komponenten durch die Testframeworks überprüft. Anschließend wird die Integration mit bestehenden Komponenten überprüft. Daran anschließend werden Serversoftware und Client-Software auf Interoperabilität geprüft. Zuletzt erfolgt die Prüfung der Funktionsweisen durch den Benutzer.

3.2.1 Testskripte

Für die Zeugniseintragefunktion, die Zeugnisvalidierungsfunktion und Zeugnisausstellfunktion sollen Testskripte zur Verfügung stehen, die diese Funktionen mit aussagekräftigen Beispieldaten testen. Für den Abnahmetest der Client-Software werden keine Testskripte benötigt, da die Funktionen einfach mithilfe von Beispielzeugnissen überprüfbar sind.

3.3 Testfälle

3.3.1 Testfall $\langle T01 \rangle$ - Sicherung der Verbindung zur Eintragekomponente durch SSL prüfen

Ziel

Überprüfung der Sicherheit der Netzwerkverbindung

Objekte/Methoden/Funktionen

<F80>,<F70>,<F60>

Pass/Fail Kriterien

Pass: Web-Browser zeigt Verbindung als sicher, Verbindung konnte aufgebaut werden
Fail: Web-Browser zeigt unsichere Verbindung beziehungsweise eine
Verbindung ohne SSL-Sicherung an, Verbindungsaufbau scheitert

Vorbedingung

Anzeigen des Web-Browsers, die die Verbindungssicherheit beschreiben. Diese sind getrennt definiert.

Einzelschritte

Web-Browser anweisen die von der Netzwerkkomponente bereitgestellte Schnittstelle aufzurufen

Beobachtungen / Log / Umgebung

Anzeigen des Web-Browsers, die die Verbindungssicherheit beschreiben. Diese sind getrennt definiert.

Besonderheiten

keine

Abhängigkeiten

keine

3.3.2 Testfall $\langle T02 \rangle$ - Eingaben der Zeugnisdaten validieren

Ziel

Überprüfung der Sicherheit, Überprüfung der Korrektheit

Objekte/Methoden/Funktionen

<RM8>

Pass/Fail Kriterien

Pass: Eingetragene Daten wurden in valides JSON übersetzt, ungültige Eingaben werden erkannt und ausgefiltert

Fail: es wurde kein Hash-Wert in die Datenbank eingetragen, ungültige Beispieldaten wurden nicht erkannt

Vorbedingung

Zeugniseintrage-Webseite wurde geöffnet, Datenbank mit noch leeren Tabellen

Einzelschritte

Felder auf der Webseite mit Beispieldaten befüllen gemäß ihrer Beschriftung, die Daten absenden indem ein Button gedrückt wird der entsprechend beschriftet ist, SQLite3 Datenbank öffnen, prüfen ob neuer Eintrag enthalten, wiederhole den kompletten Vorgang mit bekannt fehlerhaften Daten

Beobachtungen / Log / Umgebung

Einträge in der Datenbank

Besonderheiten

benötigt Namen der Datenbanktabelle, in welcher die Hash-Werte abgelegt sind

Abhängigkeiten

<T07> SecureID und Hash-Wert in Datenbank abspeichern

<T02> Eingaben für neues Zeugnis aus der Anfrage auslesen

<T08> gegenüber der <RM8, RS5> Zeugnisverwaltungskomponente authentifizieren

3.3.3 Testfall $\langle T03 \rangle$ - Eingaben der Zeugnisdaten validieren

Ziel

Überprüfung der Zuverlässigkeit

Objekte/Methoden/Funktionen

<RM8, RS5> Zeugnisverwaltungskomponente

Pass/Fail Kriterien

Pass: Es werden nur valide Zeugnisdaten akzeptiert, also solche die dem Zeugnisformat entsprechen.

Fail: Ein oder mehrere Zeugnisdaten werden akzeptiert, obwohl sie ungültig sind.

Vorbedingung

Eingabe-Webseite im Web-Browser geöffnet, Debug-Modus der <RM8, RS5> Zeugnisverwaltungskomponente ist aktiv

Einzelschritte

falsche Beispieldaten in die Eingabe-Webseite eingeben, Daten absenden, Ergebnis der Prüfung aus den Debug-Ausgaben ablesen, mit korrekten Daten wiederholen

Beobachtungen / Log / Umgebung

Debug-Ausgaben der <RM8, RS5> Zeugnisverwaltungskomponente

Besonderheiten

keine

Abhängigkeiten

<T02> Eingaben für neues Zeugnis aus der Anfrage auslesen

3.3.4 Testfall ⟨$T04$⟩ - Hash-Wert-Bildung des Zeugnisformats

Ziel

Überprüfung der Zuverlässigkeit

Objekte/Methoden/Funktionen

<F50> Hash validieren

Pass/Fail Kriterien

Pass: Für die gleichen Daten werden stets die gleichen Hash-Werte erzeugt.

Fail: Für einen oder mehrere Datensatz werden verschiedene Datensätze eingegeben.

Vorbedingung

Eingabe-Webseite im Web-Browser geöffnet, Debug-Modus der <RM8, RS5> Zeugnisverwaltungskomponente ist aktiv

Einzelschritte

Beispieldaten in die Eingabe-Webseite eingeben, Daten absenden, erzeugten Hash-Wert in der Ausgabe des Debug-Modus auslesen und notieren, den Vorgang wiederholen

Beobachtungen / Log / Umgebung

Debug-Ausgaben der <RM8, RS5> Zeugnisverwaltungskomponente

Besonderheiten

keine

Abhängigkeiten

<T02> Eingaben für neues Zeugnis aus der Anfrage auslesen

<T03> Eingaben der Zeugnisdaten validieren

3.3.5 Testfall $\langle T05 \rangle$ - Hash-Wert-Bildung des Zeugnisformats

Ziel

Überprüfung der Zuverlässigkeit der Zeugnisverwaltungskomponente

Objekte/Methoden/Funktionen

<F50> Hash validieren

Pass/Fail Kriterien

Pass: keine gleichen Hash-Werte für unterschiedliche Daten erzeugt

Fail: unterschiedliche Hash-Werte für gleiche Daten erzeugt

Vorbedingung

Web-Browser hat Eingabewebseite aufgerufen, leere Datenbanktabelle zur Speicherung der Hash-Werte

Einzelschritte

Eingabewebseite mit gültigen Beispieldaten befüllen und Daten absenden, in sqlite3 die entsprechende Tabelle öffnen, Hash-Wert notieren, wiederhole die Prozedur mit den selben Eingabedaten

Beobachtungen / Log / Umgebung

Datenbanktabelle

Besonderheiten

keine

Abhängigkeiten

<T02> Eingaben für neues Zeugnis aus der Anfrage auslesen

<T03> Eingaben der Zeugnisdaten validieren

<T07> SecureID und Hash-Wert in Datenbank abspeichern

<T08> gegenüber der <RM8, RS5> Zeugnisverwaltungskomponente authentifizieren

3.3.6 Testfall ⟨T06⟩ - SecureID auf Zufälligkeit prüfen

Ziel

Überprüfung der Sicherheit

Objekte/Methoden/Funktionen <RM8, RS5> Zeugnisverwaltungskomponente, Eingabewebseite

Pass/Fail Kriterien

Pass: SecureIDs sind annähernd zufällig verteilt

Fail: SecureIDs sind nicht annähernd zufällig verteilt

Vorbedingung

Datenbank mit leerer Tabelle für Speicherung der Werte, Webbrowser auf Eingabewebseite geöffnet

Einzelschritte

eine statistisch aussagekräftige Menge von Beispieldaten eingeben und absenden, aus der Datenbanktabelle die SecureIDs kopieren und mit geeigneter statistischer Methode auf Zufälligkeit überprüfen

Beobachtungen / Log / Umgebung

Datenbanktabellen

Besonderheiten

keine

Abhängigkeiten

<T02> Eingaben für neues Zeugnis aus der Anfrage auslesen

<T03> Eingaben der Zeugnisdaten validieren

<T07> SecureID und Hash-Wert in Datenbank abspeichern

<T08> gegenüber der <RM8, RS5> Zeugnisverwaltungskomponente authentifizieren

3.3.7 Testfall $\langle T07 \rangle$ - SecureID und Hash-Wert in Datenbank abspeichern

Ziel

Überprüfung der Zuverlässigkeit

Objekte/Methoden/Funktionen

<RM8, RS5> Zeugnisverwaltungskomponente, Eingabewebseite

Pass/Fail Kriterien

Pass: neues SecureID-Hash-Paar in Datenbank

Fail: kein SecureID-Hash-Paar in Datenbank

Vorbedingung

Datenbank mit leerer Tabelle für Speicherung der Werte, Webbrowser auf Eingabewebseite geöffnet

Einzelschritte

Beispieldaten eingeben und absenden, in SQLite3 entsprechende Tabelle betrachten

Beobachtungen / Log / Umgebung

Datenbanktabelle

Besonderheiten

keine

Abhängigkeiten

<T02> Eingaben für neues Zeugnis aus der Anfrage auslesen

<T03> Eingaben der Zeugnisdaten validieren

<T07> SecureID und Hash-Wert in Datenbank abspeichern

<T08> gegenüber der <RM8, RS5> Zeugnisverwaltungskomponente authentifizieren

3.3.8 Testfall $\langle T08 \rangle$ - gegenüber der <RM8, RS5> Zeugnisverwaltungskomponente authentifizieren

Ziel

Überprüfung der Sicherheit

Objekte/Methoden/Funktionen

<RM8, RS5> Zeugnisverwaltungskomponente, Eingabewebseite

Pass/Fail Kriterien

Pass: die Eingabe gelingt bei korrektem Login, die Eingabe scheitert bei inkorrektem Login

Fail: die Eingabe gelingt mit inkorrektem Login, die Eingabe scheitert mit korrektem Login

Vorbedingung

Verbindung zum Authentifizierungsbackend hergestellt, Web-Browser geöffnet

Einzelschritte

Eingabewebseite öffnen, mit Login-Daten anmelden

Beobachtungen / Log / Umgebung

Ausgabe des Web-Browsers

Besonderheiten

Authentifizierungsbackend benötigt

Abhängigkeiten

<T01> Sicherung der Verbindung zur Eintragekomponente durch SSL prüfen

<T10> PDF-Datei herunterladen

3.3.9 Testfall $\langle T09 \rangle$ - Zeugnis ausstellen

Ziel

Überprüfung der Funktionalität

Objekte/Methoden/Funktionen

<RM8, RS5> Zeugnisverwaltungskomponente, <RM6, RM7, RC6, RC7> Zeugnisverwaltungskomponente, Eingabewebseite

Pass/Fail Kriterien

Pass: die PDF-Datei entspricht der Formatvorlage für das Zeugnis und enthält die eingegebenen Daten

Fail: die PDF-Datei widerspricht der Formatvorlage oder enthält fehlerhafte Daten

Vorbedingung

Eingabewebseite geöffnet

Einzelschritte

Beispieldaten eintragen, Daten absenden, auf Antwort warten, warten bis das PDF geladen ist, PDF-Datei im PDF-Betrachter betrachten

Beobachtungen / Log / Umgebung

Webbrowser, PDF-Betrachter

Besonderheiten

ein PDF-Betrachter wird benötigt

Abhängigkeiten

<T02> Eingaben für neues Zeugnis aus der Anfrage auslesen

<T03> Eingaben der Zeugnisdaten validieren

<T08> gegenüber der <RM8, RS5> Zeugnisverwaltungskomponente authentifizieren

3.3.10 Testfall $\langle T10 \rangle$ - PDF-Datei herunterladen

Ziel

Überprüfung der Funktion

Objekte/Methoden/Funktionen

<RM6, RM7, RC6, RC7> Zeugnisausstellkomponente

Pass/Fail Kriterien

Pass: PDF-Datei heruntergeladen

Fail: PDF-Datei nicht oder unvollständig heruntergeladen

Vorbedingung

Eintragewebseite im Web-Browser geöffnet

Einzelschritte

Beispieldaten eintragen, Daten abschicken, Beginn des Download abwarten

Beobachtungen / Log / Umgebung

Web-Browser

Besonderheiten

keine

Abhängigkeiten

<T09> Zeugnis ausstellen

3.3.11 Testfall $\langle T11 \rangle$ - Fehlgeschlagene Verifikationsversuche beobachten

Ziel

Überprüfung der Sicherheit

Objekte/Methoden/Funktionen

<F110>

Pass/Fail Kriterien

Pass: gescheiterte Verifikationsversuche wurden alle erkannt (richtige SecureID und Hash-Wert),

Fail: ein gescheiterter Verifikationsversuch wurde nicht erkannt

Vorbedingung

Web-Browser geöffnet

Einzelschritte

Verifikationsschnittstelle mit SecureID und Hash-Wert im Web-Browser aufrufen, Antwort abwarten, Vorgang wiederholen mit falscher SecureID und falscher Hash-ID, mit richtiger SecureID und falschem Hash-Wert, mit richtigem Hash-Wert und falscher SecureID, sowie mit richtiger SecureID und richtigem Hash-Wert

Beobachtungen / Log / Umgebung

Log-Datei der <RM9> Verifikationskomponente, Web-Browser

Besonderheiten

Log-Datei der <RM9> Verifikationskomponente benötigt

Abhängigkeiten

<T12> Verbinden mit der Netzwerkkomponente zur Client-Software

<T13> Korrektes Auslesen der Client-Anfrage

<T14> Validierung der SecureID-Hash-Wert-Paare

3.3.12 Testfall $\langle T12 \rangle$ - Verbinden mit der Netzwerkkomponente zur Client-Software

Ziel

Überprüfung der Funktion

Objekte/Methoden/Funktionen <RM10> Netzwerkkomponente zur Client-Software

Pass/Fail Kriterien

Pass: Verbindung zur <RM10> Netzwerkkomponente zur Client-Software konnte hergestellt werden

Fail: Herstellung einer Verbindung zur <RM10> Netzwerkkomponente zur Client-Software scheitert

Vorbedingung

Web-Browser ist auf einem Testgerät geöffnet

Einzelschritte

mit dem Web-Browser die URL der <RM10> Verbinden mit der Netzwerkkomponente zur Client-Software aufrufen

Beobachtungen / Log / Umgebung

Fehlermeldungen des Web-Browsers

Besonderheiten

keine

Abhängigkeiten

keine

3.3.13 Testfall $\langle T13 \rangle$ - Korrektes Auslesen der Client-Anfrage

Ziel

Überprüfung der Korrektheit

Objekte/Methoden/Funktionen

<RM9> Verifikationskomponente

Pass/Fail Kriterien

Pass: alle Beispielanfragen werden korrekt erkannt

Fail: ein Beispieldatensatz wird irrtümlich als gültig erkannt, oder irrtümlich als ungültig erkannt

Vorbedingung

Web-Browser geöffnet, <RM10> Netzwerkkomponente zur Client-Software ist im Debug-Modus

Einzelschritte

URL der <RM10> Netzwerkkomponente zur Client-Software mit Beispieldaten als Argumenten aufrufen, dabei sollen die Beispieldaten für die Schnittstelle gültige URLs enthalten und auch solche die ungültig sind

Beobachtungen / Log / Umgebung

Web-Browser zur Darstellung der Antworten, Debug-Ausgaben der <RM10> Netzwerkkomponente zur Client-Software zur Darstellung der ausgelesenen Daten

Besonderheiten

keine

Abhängigkeiten

<T12> Verbinden mit der Netzwerkkomponente zur Client-Software

3.3.14 Testfall $\langle T14 \rangle$ - Validierung der SecureID-Hash-Wert-Paare

Ziel

Überprüfung der Korrektheit

Objekte/Methoden/Funktionen

<F110> Hash validieren

Pass/Fail Kriterien

Pass: alle SecureID-Hash-Wert-Paare wurden korrekt überprüft, alle nicht vorhandenen SecureIDs wurden als solche erkannt. Alle Hash-Werte, zu denen die angegebene SecureID nicht passt, wurden als solche erkannt. Die Rückmeldung an die Client-Software entspricht der Korrektheit der gesendeten Paare.

Fail: ein nicht korrektes SecureID-Hash-Wert-Paar wurde nicht als ein solches erkannt.

Vorbedingung

Debug-Modus der <RM9> Validierungskomponente ist aktiv, Web-Browser ist geöffnet, Test-Datenbank mit korrektem Teil der Beispieldaten ist geladen

Einzelschritte

URL der Netzwerkschnittstelle der <RM10> Netzwerkkomponente zur Client-Software mit Beispieldaten als Argumenten öffnen, Antwort abwarten, Antwort mit Korrektheit der gesendeten Daten vergleichen

Beobachtungen / Log / Umgebung

Debug-Ausgaben der <RM9> Verifikationskomponente, Anzeigen des Web-Browsers

Besonderheiten

die Test-Datenbank wird benötigt

Abhängigkeiten

<T12> Verbinden mit der Netzwerkkomponente zur Client-Software

<T13> Korrektes Auslesen der Client-Anfrage

3.3.15 Testfall $\langle T15 \rangle$ - Sicherheit des Validierungsmechanismus überprüfen

Ziel

Überprüfung der Sicherheit

Objekte/Methoden/Funktionen

<F110>

Pass/Fail Kriterien

Pass: keine der Beispieldaten verändert die Datenbank

Fail: ein Teil der Beispieldaten verändert die Datenbank

Vorbedingung

Web-Browser geöffnet, Test-Datenbank geladen

Einzelschritte

URL der <RM10> Netzwerkkomponente zur Client-Software mit Beispieldaten als Argumente aufrufen, Datenbank auf Änderungen prüfen, indem sie mit der Kopie der Testdatenbank verglichen wird

Beobachtungen / Log / Umgebung

Datenbank-Betrachter für SQLite3 Datenbanken

Besonderheiten

keine

Abhängigkeiten

<T12> Verbinden mit der Netzwerkkomponente zur Client-Software

<T13> Korrektes Auslesen der Client-Anfrage

<T14> Validierung der SecureID-Hash-Wert-Paare

3.3.16 Testfall $\langle T16 \rangle$ - Zeugnisse verwalten

Ziel

Überprüfung der Funktion

Objekte/Methoden/Funktionen

<RS5>, <RM8>

Pass/Fail Kriterien

Pass: alle vorhandenen Einträge können gelöscht werden

Fail: ein Eintrag konnte nicht gelöscht werden

Vorbedingung

Web-Browser geöffnet

Einzelschritte

Für jede Beispieldaten: die URL der Zeugnisverwaltungskomponente aufrufen, trage Beispieldaten ein um ein neues Zeugniserstellen zu lassen, notiere SecureID, Übersicht ausgestellter Zeugnisse anzeigen lassen, SecureID auswählen, Eintrag löschen

Beobachtungen / Log / Umgebung

Web-Browser mit Ausgaben der Verwaltungswebseite

Besonderheiten

keine

Abhängigkeiten

<T09> Zeugnis ausstellen

<T02> Eingaben für neues Zeugnis aus der Anfrage auslesen

<T07> SecureID und Hash-Wert in Datenbank abspeichern

3.3.17 Testfall $\langle T17 \rangle$ - Anwendungsschnittstelle zur Verwaltung von Zeugnissen überprüfen

Ziel

Überprüfung der Funktion

Objekte/Methoden/Funktionen

<RS5>, <RM8>

Pass/Fail Kriterien

Pass: Aktionen an die Schnittstelle werden ebenso behandelt, wie Aktionen die über die Webseite gemacht wurden

Fail: zumindest eine Aktion schlägt fehl

Vorbedingung

Web-Browser geöffnet, Test-Datenbank vorhanden

Einzelschritte

URL der <RS5, RM8> Netzwerkkomponente zur Verwaltung von Zeugnissen mit Beispieldaten zur Erstellung neuer Zeugnisse als Argumente aufrufen, Antwort mit erwartetem Ergebnis vergleichen, Löschaktion über die URL aufrufen, Antwort mit erwartetem Ergebnis vergleichen

Beobachtungen / Log / Umgebung

analog zu den Testfällen <T12> bis <T16>

Besonderheiten

keine

Abhängigkeiten

analog zu den Testfällen <T12> bis <T16>

3.3.18 Testfall $\langle T18 \rangle$ - Anfragen an den die Serversoftware haben eine maximale Wartezeit von 30 Sekunden

Ziel

Überprüfung der Benutzbarkeit

Objekte/Methoden/Funktionen

<F110>, <F60>, <F70>

Pass/Fail Kriterien

Pass: Alle gestellten Anfragen an den Server führen innerhalb von 30 Sekunden zur Anzeige eines Ergebnisses in der Client-Software.

Fail: Zumindest eine Anfrage an den Server führt innerhalb von 30 Sekunden nicht zur Anzeige eines Ergebnisses.

Vorbedingung

Die Client-Software zeigt die Auswahl zum Absenden einer Anfrage an die Serversoftware an.

Einzelschritte

die Client-Software veranlassen eine Anfrage an die Serversoftware zu senden und gleichzeitig die Zeitmessung starten, die Antwort der Serversoftware abwarten, bei Erhalt der Antwort der Serversoftware die Zeitmessung beenden

Beobachtungen / Log / Umgebung

Anzeige des Zeitmessgeräts

Besonderheiten

benötigt ein Zeitmessgerät

Abhängigkeiten

<T12> Verbinden mit der Netzwerkkomponente zur Client-Software

<T13> Korrektes Auslesen der Client-Anfrage

<T14> Validierung der SecureID-Hash-Wert-Paare

3.3.19 Testfall $\langle T19 \rangle$ - Bild des Zeugnisses aufnehmen

Ziel

Überprüfung der Funktion

Objekte/Methoden/Funktionen

<F10> Foto aufnehmen

Pass/Fail Kriterien

Pass: Es wurde ein erkennbares Bild aufgenommen.

Fail: Es wurde kein erkennbares Bild oder schlecht erkennbares Bild aufgenommen.

Vorbedingung

Die Client-Software ist im Bildaufnahmemodus. Die Umgebung ist mit Tageslicht beleuchtet.

Einzelschritte

5 Testpersonen finden, für alle Testpersonen: Bild aufnehmen, Bild anzeigen lassen, Bild manuell auf Erkennbarkeit prüfen.

Beobachtungen / Log / Umgebung

manuelle Überprüfung, Anzeige des Bildes in der Client-Software

Besonderheiten

manuelle Überprüfung, abhängig von persönlicher Wahrnehmung des Testenden

Abhängigkeiten

keine

3.3.20 Testfall $\langle T20 \rangle$ - Ausführen der Texterkennung

Ziel

Überprüfung Zuverlässigkeit

Objekte/Methoden/Funktionen

<F30> Texterkennung

Pass/Fail Kriterien

Pass: In mehr als sieben von zehn Fällen kann das Zeugnis korrekt gelesen werden.

Fail: Das Zeugnis kann in mehr zwei von zehn Fällen nicht erkannt.

Vorbedingung

Ein Bild von einem korrekten Zeugnis wurde in der Client-Software aufgenommen. Der Debug-Modus der <RM2> Texterkennungskomponente ist aktiv.

Einzelschritte

Über die <RM5> Eingabekomponente die Texterkennung starten, das Ergebnis der Texterkennung durch die <RM5> Eingabekomponente anzeigen lassen, das Ergebnis mit den Daten des Zeugnisses vergleichen

Beobachtungen / Log / Umgebung

Ausgaben der <RM5> Eingabekomponente

Besonderheiten

manuelle Überprüfung

Abhängigkeiten

<T20> Bild des Zeugnisses aufnehmen

3.3.21 Testfall $\langle T21 \rangle$ - Überprüfung der Darstellung des Ergebnis der Texterkennung

Ziel

Überprüfung Zuverlässigkeit

Objekte/Methoden/Funktionen

<F40> Text anzeigen

Pass/Fail Kriterien

Pass: Dargestelltes Ergebnis stimmt mit erkannten Daten überein.

Fail: Dargestelltes Ergebnis stimmt nicht mit erkannten Daten überein.

Vorbedingung

Ein Bild von einem korrekten Zeugnis wurde in der Client-Software aufgenommen. Der Debug-Modus der <RM2> Texterkennungskomponente ist aktiv.

Einzelschritte

Über die <RM5> Eingabekomponente die Texterkennung starten, das Ergebnis der Texterkennung durch die <RM5> Eingabekomponente anzeigen lassen. Das Ergebnis mithilfe der Debug-Ausgaben mit den Daten des Zeugnisses vergleichen, die erkannt wurden.

Beobachtungen / Log / Umgebung

Ausgaben des Debug-Modus des <RM2> Texterkennungskomponente

Besonderheiten

manuelle Überprüfung

Abhängigkeiten

<T20> Bild des Zeugnisses aufnehmen

<T22> Ausführen der Texterkennung

3.3.22 Testfall $\langle T22 \rangle$ - Erkannte Zeugnisdaten manuell korrigieren

Ziel

Überprfung der Funktion

Objekte/Methoden/Funktionen

<RM5> Eingabekomponente, <F40> Text anzeigen

Pass/Fail Kriterien

Pass: Alle erkannten Daten sind änderbar vor Absendung an die Serversoftware.

Fail: Eine oder mehrere Daten sind nicht mehr änderbar.

Vorbedingung

Es wurde ein Bild aufgenommen und darauf eine Texterkennung gemacht. Es wird das Ergebnis zur manuellen Korrektur angezeigt.

Einzelschritte

Notiere das Ergebnis der Texterkennung. Für jedes Feld mit Daten: Wähle das Feld aus und ändere den Inhalt. Überprüfe, ob die Daten geändert wurden.

Beobachtungen / Log / Umgebung

Ausgaben der <RM5> Eingabekomponente

Besonderheiten

manuelle Überprüfung

Abhängigkeiten

<T20> Bild des Zeugnisses aufnehmen

<T22> Ausführen der Texterkennung

3.3.23 Testfall $\langle T23 \rangle$ - Erzeugen des Zeugnisformats

Ziel

Überprüfung der Zuverlässigkeit

Objekte/Methoden/Funktionen

<F30> Texterkennung

Pass/Fail Kriterien

Pass: Alle Zeugnisdaten wurden korrekt übersetzt.

Fail: Ein oder mehrere Zeugnisdaten wurden nicht in das korrekte Format übersetzt.

Vorbedingung

Es wurde ein Bild aufgenommen und darauf eine Texterkennung gemacht. Der Debug-Modus der <RM2> Texterkennungskomponente ist aktiv.

Einzelschritte

in der Ausgabe des Debug-Modus die formatierten Zeugnisdaten auslesen und einen Syntaxcheck mit dem Zeugnisformat machen.

Beobachtungen / Log / Umgebung

Ausgaben des Debug-Modus, Zeugnisformat

Besonderheiten

manuelle Überprüfung

Abhängigkeiten

<T20> Bild des Zeugnisses aufnehmen

<T22> Ausführen der Texterkennung

3.3.24 Testfall $\langle T24 \rangle$ - Sicherheitsüberprüfung der Kommunikation zwischen Client-Software und Serversoftware

Ziel

Überprüfung der Sicherheit

Objekte/Methoden/Funktionen

<F60>, <F70>

Pass/Fail Kriterien

Pass: Die Verbindung ist durch SSL gesichert.

Fail: Die Verbindung ist unzureichend durch SSL gesichert oder nicht gesichert.

Vorbedingung

Client-Software befindet sich in der Auswahl, ob das ermittelte SecureID-Hash-Wert-Paar zur Überprüfung an die Serversoftware gesendet werden kann. Ein Packet-Analyser läuft und kann Pakete der Client-Software mitlesen.

Einzelschritte

Aufnahme im Packet-Analyser starten, Client-Software veranlassen das SecureID-Hash-Wert-Paar an die Serversoftware zu senden, im Packet-Analyser überprüfen, ob an die Adresse der Server-Software von der Client-Software Pakete mit SSL-Transportwegsicherung gesendet wurden.

Beobachtungen / Log / Umgebung

Ausgaben des Packet-Analysers

Besonderheiten

benötigt einen Packet-Analyser

Abhängigkeiten

keine

3.3.25 Testfall $\langle T25 \rangle$ - Darstellung der Validierungsantwort überprüfen

Ziel

Überprüfung der Zuverlässigkeit

Objekte/Methoden/Funktionen

<F110> Hash validieren

Pass/Fail Kriterien

Pass: die angezeigte Validierungsantwort stimmt inhaltlich in allen Fällen mit der tatsächlichen Antwort überein

Fail: die angezeigte Validierungsantwort stimmt in einem oder mehreren Fällen nciht mit der tatsächlichen Antwort überein

Vorbedingung

Die <RM10> Netzwerkkomponente zur Client-Software befindet sich im Debug-Modus.

Einzelschritte

Sende mithilfe der Client-Software für eine Menge an Beispielzeugnissen Validierungsanfragen an die <RM10> Netzwerkkomponente zur Client-Software, vergleiche die Validierungsantwort in der Debug-Ausgabe der <RM10> Netzwerkkomponente zur Client-Software mit der in der <RM5> Eingabekomponente angezeigten Validierungsantwort

Beobachtungen / Log / Umgebung

Debug-Ausgaben der <RM10> Netzwerkkomponente zur Client-Software, Ausgaben der <RM5> Eingabekomponente

Besonderheiten

keine

Abhängigkeiten

<T12> Verbinden mit der Netzwerkkomponente zur Client-Software <T13> Korrektes Auslesen der Client-Anfrage

<T14> Validierung der SecureID-Hash-Wert-Paare

3.3.26 Testfall $\langle T26 \rangle$ - Test der Client-Software auf unterschiedlichen Gerätetypen

Ziel

Überprüfung der Funktion

Objekte/Methoden/Funktionen

<RM1>, <RC2>, <RM2>, <RM3>, <AM4>, <RM5> Client-Software

Pass/Fail Kriterien

Pass: Alle Funktionen der Client-Software stehen auf allen Testgeräten bereit.

Fail: Eine oder mehrere Funktionen der Client-Software stehen nicht auf allen Testgeräten bereit.

Vorbedingung

Auf allen Testgeräten ist die Client-Software gestartet.

Einzelschritte

Durchführung aller die Client-Software betreffenden Testfälle auf jedem der Testgeräte.

Beobachtungen / Log / Umgebung

analog zu jeweiligem Testfall

Besonderheiten

Sammlung von Testfällen

Abhängigkeiten

analog zu jeweiligem Testfall

3.3.27 Testfall $\langle T27 \rangle$ - Bereits aufgenommene Fotos überprüfen

Ziel

Überprüfung der Funktion

Objekte/Methoden/Funktionen

<F20>

Pass/Fail Kriterien

Pass: Es kann ein Bild aus dem Photoalbum ausgewählt werden, um dies an die Texterkennungskomponente zu übergeben.

Fail: Es kann kein Bild aus dem Photoalbum ausgewählt werden, um dies an die Texterkennungskomponente zu übergeben.

Vorbedingung

Die Client-Software ist gestartet.

Einzelschritte

Das Photoalbum aus der Client-Software heraus öffnen und ein Bild auswählen.

Beobachtungen / Log / Umgebung

Ausgaben der <RM5> Eingabekomponente

Besonderheiten

manuelle Überprüfung

Abhängigkeiten

keine

4 Integrationstest

Im folgenden werden die verschiedenen Testfälle der Integrationstests beschrieben. Ziel dieser Tests ist es, die durch die Unit-Tests getesteten Funktionen auf ihr Zusammenspiel zu überprüfen. Dabei wird einmal darauf eingegangen wie die unterschiedlichen Komponenten der App und des Servers untereinander interagieren und wie der Server mit der App kommuniziert. Nach dem durchführen dieser Tests sollte gewehrleistet sein, dass alle Funktionen gut zusammen arbeiten.

4.1 Zu testende Komponenten

In der folgenden Tabelle werden die jeweiligen Komponenten zusammen mit ihren Testfällen aufgelistet.

Nr	Komponenten	Testfälle	Kommentar
1	<C10> Image + <C20> ImageProcessing	<T1500>, <T2000>	-
2	<C20> ImageProcessing + <C50> Data + <C30> Verfification	<T3000>	-
3	<C30> Verification + <C40> Communication	<T5000>	-
4	<C40> Communication + <C80> Verification	<T4000>	-
5	<C10> Image + <C20> ImageProcessing + <C30> Verification + <C40> Communication + <C50> Data + <C80> Verification	<T6000>	-
6	<C70> Views + <C120> Rest-API	<T7000>, <T8000>, <T9000>, <T10000>, <T12000>	Diese Tests sollen die Kompatibilität der Webseite und der API sicherstellen.

7	<C90> Models + <C80> Verification	<T13000>	Dieser Test überprüft die Überprüfbarkeit von eingetragenen Zeugnissen

4.2 Testverfahren

Die Komponenten <C10> bis <C50>, aus denen die App besteht werden alle ohne Testskripte getestet. Der Grund dafür ist, dass sich das korrekte Zusammenspiel der Komponenten viel besser von einem Tester manuell unter realen Bedinungen überprüfen lässt, als durch ein Skript mit "Mockup-Komponenten". Die Komponenten <C70> Views, <C80> Verification, und <C120> Rest-API werden durch <C130> Django in Schnittstellen nach aussen eingebunden. Die Schnittstellen, die innerhalb des Servers zwischen diesen Komponenten und auch <C90> Models bestehen, werden ausschließlich durch <C130> Django, <C110> Texlive und <C100> Database bereitgestellt und sind deshalb durch Tests dieser Projekte abgedeckt. Deshalb werden diese Schnittstellen hier nicht gesondert getestet. Die selbst erstellten Komponenten können also, indem diese Schnittstellen als Interfaces genutzt werden, miteinander auf Integration betestet werden. Da aufgrund der hohen Verknüpftheit der Komponenten miteinander ein Test nach der Bottom-Up oder der Top-Down-Methode einen zu großen Testaufwand ergeben hätte, wurde hier die Big-Bang Testmethode gewählt, welche alle bisher erstellten Komponenten gleichzeitig testet. Es wurde dabei in den unterschiedlichen Tests getrennt auf unterschiedliche Komponenten eingegangen.

4.2.1 Testskripte

Für diese Testfälle werden keine Testskripte verwendet.

4.3 Testfälle

4.3.1 Testfall $\langle T1500 \rangle$ - Komponente <C10> + <C20>

Ziel

Ziel dieses Testes ist es zu überprüfen, ob die Komponenten <C10> Image das in ihr erzeugte Bild korrekt an die Komponenten <C20> ImageProcessing übergibt, wenn der Nutzer selber ein Bild aufnimmt.

Objekte/Methoden/Funktionen

Die Relevanten Klassen für diesen Test sind: Camera2Fragment, ProcessingActivity
Im Camera2Fragment wird der Intent zum starten der ProcessingActivity erzeugt. Hierbei wird auch das Bild als extra übergeben.

Pass/Fail Kriterien

Pass: In der ImageProcessingActivity wird das aufgenommene Bild angezeigt.
Fail: Es wird kein Bild angezeigt oder es tritt ein anderer Fehler auf.

Vorbedingung

-

Einzelschritte

Der Tester macht durch das Drücken auf den Auslöser ein Bild, nachdem die App gestartet wurde. Danach wird das Bild automatisch an das ImageProcessing übergeben, das heißt es lässt sich direkt erkennen ob der Test fehlgeschalgen ist.

Beobachtungen / Log / Umgebung

Wie in den Einzelschritten zu sehen, ist das Ergebnis des Testes unmittelbar auf dem Bildschirm zu erkennen. Für weitere Informationen kann der Tester LogCat nutzen.

Besonderheiten

Dieser Test wird ohne Testskript ausgeführt.

Abhängigkeiten

-

4.3.2 Testfall $\langle T2000 \rangle$ - Komponente <C10> + <C20>

Ziel

Ziel dieses Testes ist es zu überprüfen, ob die Komponenten <C10> Image das in ihr erzeugte Bild korrekt an die Komponenten <C20> ImageProcessing übergibt, wenn der Nutzer ein Bild aus der Galarie wählt.

Objekte/Methoden/Funktionen

Die Relevanten Klassen für diesen Test sind: GalaryStarterActivity, ProcessingActivity
In der GalaryStarterActivity wird der Intent zum starten der ProcessingActivity erzeugt. Hierbei wird auch das Bild als extra übergeben.

Pass/Fail Kriterien

Pass: In der ImageProcessingActivity wird das ausgewählte Bild angezeigt.
Fail: Es wird kein Bild angezeigt oder es tritt ein anderer Fehler auf.

Vorbedingung

-

Einzelschritte

Der Tester wählt ein Bild aus der Galarie aus, in dem er erst auf den Galarie Button drückt und dann ein Bild auswählt. Anschließend wird das Bild direkt an das ImageProcessing übergeben und es lässt sich sofort erkennen ob der Test fehlgeschlagen ist.

Beobachtungen / Log / Umgebung

Wie in den Einzelschritten zu sehen, ist das Ergebnis des Testes unmittelbar auf dem Bildschirm zu erkennen. Für weitere Informationen kann der Tester LogCat nutzen.

Besonderheiten

Dieser Test wird ohne Testskript ausgeführt.

Abhängigkeiten

-

4.3.3 Testfall $\langle T3000 \rangle$ - Komponente <C20> + <C50> + <C30>

Ziel

Ziel dieses Testes ist es, zu überprüfen, ob aus dem im <C20> ImageProcessing erzeugten Bitmap in der <C50> Data-Komponente ein Certificate erzeugt wird, dass anschließend an die Komponente <C30> Verification übergeben wird.

Objekte/Methoden/Funktionen

In der `ProcessingActivity` ist die Subklasse `OCRTask` dafür zuständig, nach dem berechnen des HOCR-Outputes, diesen an die `Certificate` Klasse der <C50> Data-Komponente zu übergeben. Dies geschieht über den Kontruktor der `Certificate` Klasse. Anschließend wird das so erzeugte `Certificate` an über einen Intent an der `ReviewActivity` übergeben.

Pass/Fail Kriterien

Pass: In der ReviewActivity werden Daten in den Textfeldern angezeigt. Fail: Alle anderen möglichen Ergebnisse.

Vorbedingung

Es muss in der `ProcessingActivity` ein Bild eines Zeugnisses vorhanden sein.

Einzelschritte

Der Tester führt durch einen Druck auf den Verify-Button in der `ProcessingActivity` das ImageProcessing und anschließend den OCR-Scan aus. Nachdem OCR-Scan wird automatisch aus den von TessTwo zurückgegebenen Daten das `Certificate` erstellt. Sollte dies fehlschlagen, so wird eine Fehlermeldung angezeigt und der Test ist damit nicht erfolgreich. Nachdem das `Certificate` erzeugt wurde, wird es automatisch an die `ReviewActivity` weitergeleitet und die Textfelder werden erzeugt.

Beobachtungen / Log / Umgebung

Wie in den Einzelschritten zu sehen, ist das Ergebnis des Testes unmittelbar auf dem Bildschirm zu erkennen. Für weitere Informationen kann der Tester LogCat nutzen.

Besonderheiten

Dieser Test wird ohne Testskript ausgeführt.

Abhängigkeiten

<T1500>, <T2000>

4.3.4 Testfall $\langle T4000 \rangle$ - Komponente <C40> + <C80>

Ziel

In diesem Test wird die Kommunikation zwischen App und Server getestet.

Objekte/Methoden/Funktionen

Die Komponenten <C40> wird in der `Communication`-Klasse eine Serververbindung herstellen und das Resultat, das von der Serverkomponenten <C80> zurückgeliefert wird verarbeiten.

Pass/Fail Kriterien

Pass: Es wird keine Fehlermeldung angezeigt. Fail: Es wird eine Fehlermeldung angezeigt.

Vorbedingung

Der `Communication`-Klasse müssen die Daten eines Zeugnisses übergeben werden.

Einzelschritte

Der Tester startet die Verification eines Zeugnisses. Die Komponente <C40> Communication wird versuchen eine Serververbindung aufzubauen und das Ergebnis des Servers zu verarbeiten. Sollte bei der Kommunikation zwischen Server und App ein Fehler auftreten und kein Ergebnis übermittelt werden oder keine Verbindung aufgebaut werden können, so wird eine Fehlermeldung angezeigt.

Beobachtungen / Log / Umgebung

Wie in den Einzelschritten zu sehen, ist das Ergebnis des Testes unmittelbar auf dem Bildschirm zu erkennen. Für weitere Informationen kann der Tester LogCat nutzen.

Besonderheiten

Dieser Testfall hängt sehr eng mit dem Testfall <T5000> zusammen und sie sollten beide zusammen ausgeführt werden. Für eine bessere Übersicht wurden sie trotzdem getrennt aufgeschrieben.

Abhängigkeiten

<T1500>, <T2000>, <T3000>, <T5000>

4.3.5 Testfall $\langle T5000 \rangle$ - Komponente <C30> + <C40>

Ziel

Ziel dieses Tests ist es, zu übeprüfen, ob die zu überprüfenden Daten in der <C30> Verification-Komponenten korrekt an die <C40> Communication Komponenten weiter gegeben werden und anschließend das Ergebniss der überprüfung wider zurück gegeben wird.

Objekte/Methoden/Funktionen

Relevant ist in der <C30> Verification-Komponente die `ReviewActivity`, die die Subklasse `CommunicationTask` startet. Hierbei wird der <C40> Communication Komponente der Inhalt des Zeugnisses übergeben. Nachdem der `CommunicationTask` ein Ergebnis hat, ruft er per Intent die `ReviewActivity` auf und übergibt ihr das Ergebnis.

Pass/Fail Kriterien

Pass: Wird ein Haken angezeigt, so war der Test erfolgreich. Fail: Tritt ein Fehler auf, oder wird ein Kreuz angezeigt so ist der Test fehlgeschlagen.

Vorbedingung

Wichtig ist, dass in der <C30> Verification-Komponenten ein valides Zeugnis vorliegt.

Einzelschritte

Der Tester drückt in der `ReviewActivity` denn Verify-Button. danach passiert automatisch die Kommunikation unter den Komponenten und der Tester kann auf dem Bildschirm das Ergebnis ablesen.

Beobachtungen / Log / Umgebung

Wie in den Einzelschritten zu sehen, ist das Ergebnis des Testes unmittelbar auf dem Bildschirm zu erkennen. Für weitere Informationen kann der Tester LogCat nutzen.

Besonderheiten

Dieser Testfall hängt sehr eng mit dem Testfall <T4000> zusammen und sie sollten beide zusammen ausgeführt werden. Für eine bessere Übersicht wurden sie trotzdem getrennt aufgeschrieben.

Abhängigkeiten

<T1000>, <T2000>, <T3000>, <T4000>

4.3.6 Testfall $\langle T6000 \rangle$ - Komponente <C10> + <C20> + <C30> + <C40> + <C80>

Ziel

Ziel dieses Tests ist zu prüfen, ob alle Komponenten der App korrekt zusammen arbeiten.

Objekte/Methoden/Funktionen

Grundsätzlich werden alle oben schon einzelln getesteten Komponenten-Integrationen zusammen geführt. Daher sind hier alle Komponenten und Objekte der App relevant.

Pass/Fail Kriterien

Pass: Am Ende der Überprüfung wird angezeigt, dass das Zeugnis valide ist. Fail: Es tritt eine Fehlermeldung auf, oder das Zeugnis wird nicht als valide erkannt.

Vorbedingung

In der Galarie des Handys befindet sich ein Zeugnis, dass vom OCR-Scan korrekt erkannt wird und valide ist.

Einzelschritte

Der Tester drückt den Galarie–Button in der `MainActivity` und wählt das Zeugniss aus. Anschließend startet er mit einem Druck auf den Next-Button das ImageProcessing und den OCR-Scan. Wird nun der Inhalt des Zegnisses in der `ReviewActivity` angezeigt, so startet er die eigentliche überprüfung mit einem druck auf den Verify-Button. Anschließend ist auf dem Bildschirm ein weißer Hacken auf grünem Grund zu sehen, wenn der Test erfolgreich war.

Beobachtungen / Log / Umgebung

Wie in den Einzelschritten zu sehen, ist das Ergebnis des Testes unmittelbar auf dem Bildschirm zu erkennen. Für weitere Informationen kann der Tester LogCat nutzen.

Besonderheiten

Dieser Test führt alle Komponenten der App und die Server-Komponente die für die Kommunikation mit der App zuständig ist zusammen. Daher solltet dieser Test erst ausgeführt werden, wenn alle anderen Tests erfolgreich waren.

Abhängigkeiten

<T2000>, <T3000>, <T4000>, <T5000>

4.3.7 Testfall $\langle T7000 \rangle$ - Komponente <C70> Views und <C120> Rest-API

Ziel

Ziel dieses Tests ist zu prüfen, ob alle Zeugnisdaten, die über die Webseite eingetragen werden auch über die API verfügbar sind.

Objekte/Methoden/Funktionen

certificates.serializers, certificates.admin, certificates.viewsets, certificates.models

Pass/Fail Kriterien

Pass: Über die Webseite getätigte Änderunge am Datenbestand können über die API abgerufen werden. Fail: Es können einzelne oder mehrere Daten nicht über die API abgerufen werden.

Vorbedingung

Der Server ist im Debug-Modus gestartet. Die Eingabewebseite ist in einem Web-Browser geöffnet. Die Web-Ansicht der API ist in einem Web-Browser geöffnet.

Einzelschritte

Der Tester wählt auf der Webseite die Option um einen Zeugnisinhaber zu erstellen aus und erstellt 5 Zeugnisinhaber.

Der Tester wählt auf der Webseite die Option um ein neues Fach anzulegen aus und legt 10 neue Fächer an.

Der Tester wählt die Option um eine neue Institution anzulegen aus und legt 10 neue Institutionen an.

Der Tester wählt die Option aus um ein neues Zeugnis anzulegen und legt 5 neue Zeugnisse mit je zumindest 3 Noten an.

Der Tester ruft die für die Schnittstellen spezifizierten URLs der API auf, um die Zeugnisdaten aufzulisten.

Beobachtungen / Log / Umgebung

Das Ergebnis des Tests lässt sich im Webbrowser ablesen.

Besonderheiten

Dieser Test erfordert manuelle Eingaben.

Abhängigkeiten

<T800> bis <T1207>

4.3.8 Testfall $\langle T8000 \rangle$ - Komponente <C70> Views und <C120> Rest-API

Ziel

Ziel dieses Tests ist zu prüfen, ob alle Zeugnisdaten, die über die Webseite geändert werden auch über die API verfügbar sind.

Objekte/Methoden/Funktionen

certificates.serializers, certificates.admin, certificates.viewsets, certificates.models

Pass/Fail Kriterien

Pass: Über die Webseite getätigte Änderunge am Datenbestand können über die API abgerufen werden. Fail: Es können einzelne oder mehrere Daten nicht über die API abgerufen werden.

Vorbedingung

Der Server ist im Debug-Modus gestartet. Die Eingabewebseite ist in einem Web-Browser geöffnet. Die Web-Ansicht der API ist in einem Web-Browser geöffnet.

Einzelschritte

Der Tester wählt auf der Webseite die Option um einen Zeugnisinhaber zu ändern aus und ändert 5 Zeugnisinhaber.

Der Tester wählt auf der Webseite die Option um ein Fach zu ändern aus und ändert 5 Fächer.

Der Tester wählt die Option um eine Institution zu ändern aus und ändert 5 Institutionen.

Der Tester wählt die Option aus um ein Zeugnis zu ändern und ändert 5 Zeugnisse mit je zumindest 3 Noten.

Der Tester ruft die für die Schnittstellen spezifizierten URLs der API auf, um die Zeugnisdaten aufzulisten.

Beobachtungen / Log / Umgebung

Das Ergebnis des Tests lässt sich im Webbrowser ablesen.

Besonderheiten

Dieser Test erfordert manuelle Eingaben.

Abhängigkeiten

<T800> bis <T1207>, <T7000>

4.3.9 Testfall $\langle T9000 \rangle$ - Komponente <C70> Views und <C120> Rest-API

Ziel

Ziel dieses Tests ist zu prüfen, ob alle Zeugnisdaten, die über die Webseite gelöscht werden auch über die API nicht mehr verfügbar sind.

Objekte/Methoden/Funktionen

certificates.serializers, certificates.admin, certificates.viewsets, certificates.models

Pass/Fail Kriterien

Pass: Über die Webseite getätigte Änderungen am Datenbestand können über die API abgerufen werden. Fail: Es können einzelne oder mehrere Daten, die gelöscht wurden über die API abgerufen werden.

Vorbedingung

Der Server ist im Debug-Modus gestartet. Die Eingabewebseite ist in einem Web-Browser geöffnet. Die Web-Ansicht der API ist in einem Web-Browser geöffnet.

Einzelschritte

Der Tester wählt die Option aus um ein Zeugnis zu löschen und löscht 5 Zeugnisse mit je zumindest 3 Noten.

Der Tester wählt auf der Webseite die Option um ein Fach zu löschen aus und löscht 5 Fächer.

Der Tester wählt die Option um eine Institution zu löschen aus und löscht 5 Institutionen.

Der Tester wählt auf der Webseite die Option um einen Zeugnisinhaber zu löschen aus und löscht 5 Zeugnisinhaber.

Der Tester ruft die für die Schnittstellen spezifizierten URLs der API auf, um die Zeugnisdaten aufzulisten.

Beobachtungen / Log / Umgebung

Das Ergebnis des Tests lässt sich im Webbrowser ablesen.

Besonderheiten

Dieser Test erfordert manuelle Eingaben.

Abhängigkeiten

<T800> bis <T1207>, <T7000>

4.3.10 Testfall $\langle T10000 \rangle$ - Komponente <C70> Views und <C120> Rest-API

Ziel

Ziel dieses Tests ist zu prüfen, ob alle Zeugnisdaten, die über die API erstellt werden auch über die Webseite verfügbar sind.

Objekte/Methoden/Funktionen

certificates.serializers, certificates.admin, certificates.viewsets, certificates.models

Pass/Fail Kriterien

Pass: Über die API getätigte Änderungen am Datenbestand können über die Webseite abgerufen werden. Fail: Es können einzelne oder mehrere Daten, die geändert wurden nicht über die Webseite abgerufen werden.

Vorbedingung

Der Server ist im Debug-Modus gestartet. Die Webansicht der API ist in einem Web-Browser geöffnet. Die Webseite ist in einem Web-Browser geöffnet.

Einzelschritte

Der Tester ruft die URL der Institutionen auf und legt, indem er JSON-formatierte Daten in das Testfeld eingibt 10 neue Institutionen an.

Der Tester ruft die URL der Zeugnisinhaber auf und legt, indem er JSON-formatierte Daten in das Testfeld eingibt 10 neue Zeugnisinhaber an.

Der Tester ruft die URL der Zeugnisfächer auf und legt, indem er JSON-formatierte Daten in das Testfeld eingibt 10 neue Zeugnisfächer an.

Der Tester ruft die URL der Zeugnisse auf und legt, indem er JSON-formatierte Daten in das Testfeld eingibt 5 neue Zeugnisse mit je zumindest 3 unterschedlichen Noten an.

Beobachtungen / Log / Umgebung

Das Ergebnis des Tests lässt sich im Webbrowser ablesen.

Besonderheiten

Dieser Test erfordert manuelle Eingaben.

Abhängigkeiten

<T800> bis <T1207>

4.3.11 Testfall $\langle T11000 \rangle$ - Komponente <C70> Views und <C120> Rest-API

Ziel

Ziel dieses Testes ist zu prüfen, ob alle Zeugnisdaten, die über die API geändert werden auch über die Webseite verfügbar sind.

Objekte/Methoden/Funktionen

certificates.serializers, certificates.admin, certificates.viewsets, certificates.models

Pass/Fail Kriterien

Pass: Über die API getätigte Änderungen am Datenbestand können über die Webseite abgerufen werden. Fail: Es können einzelne oder mehrere Daten, die geändert wurden nicht über die Webseite abgerufen werden.

Vorbedingung

Der Server ist im Debug-Modus gestartet. Die Webansicht der API ist in einem Web-Browser geöffnet. Die Webseite ist in einem Web-Browser geöffnet.

Einzelschritte

Der Tester ruft je die URL einer Institution auf und ändert, indem er JSON-formatierte Daten in das Testfeld eingibt 10 Institutionen.

Der Tester ruft je die URL eines Zeugnisinhbers auf und ändert, indem er JSON-formatierte Daten in das Testfeld eingibt 10 Zeugnisinhaber.

Der Tester ruft je die URL eines Zeugnisfaches auf und ändert, indem er JSON-formatierte Daten in das Testfeld eingibt 10 Zeugnisfächer an.

Der Tester ruft je die URL eines Zeugniss auf und ändert, indem er JSON-formatierte Daten in das Testfeld eingibt 5 Zeugnisse mit je zumindest 3 unterschedlichen Noten.

Beobachtungen / Log / Umgebung

Das Ergebnis des Tests lässt sich im Webbrowser ablesen.

Besonderheiten

Dieser Test erfordert manuelle Eingaben.

Abhängigkeiten

<T800> bis <T1207>, <T10000>

4.3.12 Testfall ⟨$T12000$⟩ - Komponente <C70> Views und <C120> Rest-API

Ziel

Ziel dieses Tests ist zu prüfen, ob alle Zeugnisdaten, die über die API gelöscht werden auch über die Webseite nicht mehr verfügbar sind.

Objekte/Methoden/Funktionen

certificates.serializers, certificates.admin, certificates.viewsets, certificates.models

Pass/Fail Kriterien

Pass: Über die API gelöschte Daten aus dem Datenbestand können über die Webseite nicht mehr abgerufen werden. Fail: Es können einzelne oder mehrere Daten, die gelöscht wurden über die Webseite abgerufen werden.

Vorbedingung

Der Server ist im Debug-Modus gestartet. Die Webansicht der API ist in einem Web-Browser geöffnet. Die Webseite ist in einem Web-Browser geöffnet.

Einzelschritte

Der Tester ruft die URL eines Zeugniss auf und löscht dieses über den dazu vorgesehenen Button.

Der Tester ruft die URL eines Zeugnisfachs auf und löscht dieses über den dazu vorgesehenen Button.

Der Tester ruft die URL eines Zeugnisinhabers auf und löscht diesen über den dazu vorgesehenen Button.

Der Tester ruft die URL einer Institution auf und löscht diese über den dazu vorgesehenen Button.

Beobachtungen / Log / Umgebung

Das Ergebnis des Tests lässt sich im Webbrowser ablesen.

Besonderheiten

Dieser Test erfordert manuelle Eingaben.

Abhängigkeiten

<T800> bis <T1207>, <T10000>

4.3.13 Testfall $\langle T13000 \rangle$ - <C90> Models und <C80> Verification

Ziel

Ziel dieses Tests ist zu prüfen, ob alle Zeugnisse, die durch die Objekte von Klassen in certificates.models dargestellt werden, durch die Verifikationsschnittstelle überprüft werden können.

Objekte/Methoden/Funktionen

certificates.serializers, certificates.admin, certificates.viewsets, certificates.models

Pass/Fail Kriterien

Pass: Alle vorhandenen Zeugnisse konnten validiert werden. Fail: Es konnten ein oder mehrere Zeugnisse nicht validiert werden.

Vorbedingung

Der Server ist im Debug-Modus gestartet.

Einzelschritte

Der Tester ruft für 5 Zeugnisse die URL der Verifikationsschnittstelle wie in der Schnittstellenspezifikation beschrieben mit der bekannten SecureID und dem bekannten Hash-Wert auf.

Beobachtungen / Log / Umgebung

Das Ergebnis des Tests lässt sich im Webbrowser ablesen. Für gelungene Validierungen wird in dem zurückgegebenen JSON-Text 'result' = true stehen.

Besonderheiten

Dieser Test erfordert manuelle Eingaben.

Abhängigkeiten

<T800> bis <T1207>, <T10000>

5 Unit-Tests

In diesem Kapitel werden die Unit-Tests für Server und App beschrieben. Ziel dieser Tests ist es, alle relevanten Funktionen einzelnen zu testen. Besonderen wert wurde dabei darauf gelegt zu testen was bei eventuellen fehleingaben oder ähnlichem passiert, um sowohl App als auch Server möglichst benutzer freundlich zu halten.

5.1 Zu testende Komponenten

In der folgenden Tabelle werden die einzelnen zu testenden Komponenten und die jeweiligen Testfälle aufgelistet.

Nr	Komponenten	Testfälle	Kommentar
1	<C10> Image	<T100> - <T101>	-
2	<C20> ImageProcessing	<T200> - <T205>	-
3	<C30> Verification	<T300> - <T307>	-
4	<C40> Communication	<T400> - <T410>	-
5	<C50> Data	<T500> - <T522>	-
6	<C60> Authentication	<T403>	Diese Komponente wird auf Seiten der App im Rahmen der `Communication` Komponente getestet
7	<C70> Views	<T700> - <T706>	-
8	<C80> Verification	<T800>, <T801>	-
9	<C90> Models	<T900>	-

10	<C120> Rest-API	<T1200> - <T1207>	GET, POST, PUT, DELETE

5.2 Testverfahren

Die Tests werden entweder manuell vom Tester durchgeführt, um möglichst unter real Bedingungen zu testen. Dies setzt allerdings voraus, dass die Testfälle der Reihe nach abgearbeitet werden da sie aufeinander aufbauen. Allerdings gibt es auch Funktionen die durch Testskripte überprüft werden. Diese lassen sich jederzeit Testen, da sie keine Abhängigkeiten auf andere Testfälle besitzen.

5.2.1 Testskripte

Die meisten Funktionen der App werden ohne Testskripte getestet, da es vor allem auf die visuelle Rückmeldung an den Nutzer ankommt die such durch Skripte nur schwer überprüfen lässt. Folgende Skripte wurden verwendet:
`CreatingTextboxesTest CreateConnectionTest CreateRequestURLTest ParseAnswerTestEmpty ParseAnswerTestPositive ImageProcessingNoPicture`
Für den Server werden überwiegend Testskripte verwendet. Die graphische Schnittstelle zu den Benutzern kann nur manuell getestet werden. Folgende Skripte wurden verwendet: `ValidationTestCase PdfExportTestCase RestApiPostTestCase RestApiGetTestCase RestApiDeleteTestCase RestApiPutTestCase SecureIDAndHashGenerationTestCase`

Diese können mit manage.py direkt von der Kommandozeile aufgerufen werden.

5.3 Testfälle

5.3.1 Testfall $\langle T100 \rangle$ - MainActivity

Ziel

Überprüfung ob die .trainddata Dateien auf den lokalen Speicher kopiert wurden.

Objekte/Methoden/Funktionen

MainActivity, onCreate(Bundle savedInstanceState), copyTessData()

Pass/Fail Kriterien

Pass: LogCat: "Kopieren war erfolgreich"

Fail: LogCat: "Beim Kopieren ist was schief gegangen"

Vorbedingung

1. Die App wurde auf einem Android Gerät installiert.
2. Die App wurde vorher noch nie gestartet.
3. Das Verzeichnis /tessdata darf auf dem Gerät noch nicht existieren.

Einzelschritte

1. Der Nutzer öffnet die App.
2. Der Nutzer überprüft die LogCat Einträge.

Beobachtungen / Log / Umgebung

Der Test wird auf einem Android fähigen Endgerät ausgeführt, als Log dient LogCat in Android Studio.

Besonderheiten

Der Test kann nur bei erstmaliger Installation durchgeführt werden.

5.3.2 Testfall $\langle T101 \rangle$ - GalerieStarterActivtity

Ziel

Überprüfen, ob die MainActivity korrekt startet, nachdem der Benutzer aus der Review-Activity zurück navigiert oder die Bilderauswahl abbricht.

Objekte/Methoden/Funktionen

MainActivity, GalerieStarterActivtity

Pass/Fail Kriterien

Pass: Die MainActivity startet normal.

Fail: Die GalerieStarterActivity startet und zeigt eine Error Nachricht an.

Vorbedingung

1. Die App wurde auf einem Android Gerät installiert.
2. Die App wurde gestartet.

Einzelschritte

1. Der Nutzer drückt auf den Galerie Button in der Actionbar.
2. Der Nutzer bricht die Bilderauswahl über den Abbrechen Button ab.

Beobachtungen / Log / Umgebung

Der Test wird auf einem Android fähigen Endgerät ausgeführt. Der Nutzer protokolliert seine Beobachtungen.

Besonderheiten

Der Test kann unvorhersehbar scheitern.

5.3.3 Testfall $\langle T200 \rangle$ - Processing Activity

Ziel

Überprüfung ob die Texterkennung korrekt in einem eigenen Thread ausgeführt wird.

Objekte/Methoden/Funktionen

Processing Activity, OCRTask, textRecognition(Bitmap pic)

Pass/Fail Kriterien

Pass: Der Thread wird korrekt im Device Monitor angezeigt.

Fail: Jedes Ergebnis was nicht dem Pass - Kriterium entspricht.

Vorbedingung

Keine.

Einzelschritte

Die App wird ordnungsgemäß gestartet und mit dem Debugger von Android Studio wärend des Image Processings angehalten. Nun kann der Tester dort kontrollieren ob der Thread korrekt angezeigt wird.

Beobachtungen / Log / Umgebung

Dieser Test wird in Android Studio ausgeführt. Als Log dient Logcat und der Device Monitor sowie der Debugger.

Besonderheiten

Dieser Test wird von keinem Skript ausgeführt, sondern manuell von einem Tester.

Abhängigkeiten

<T204>, <T205>

5.3.4 Testfall $\langle T201 \rangle$ - Processing Activity

Ziel

Überprüfung ob die Bildbearbeitung korrekt in einem eigenen Thread ausgeführt wird.

Objekte/Methoden/Funktionen

Processing Activity, ImageProcessingTask

Pass/Fail Kriterien

Pass: Der Thread wird korrekt im Device Monitor angezeigt.

Fail: Jedes Ergebnis was nicht dem Pass - Kriterium entspricht.

Vorbedingung

Keine.

Einzelschritte

Die App wird ordnungsgemäß gestartet und mit dem Debugger von Android Studio wärend der Texterkennung angehalten. Nun kann der Tester dort kontrollieren ob der Thread korrekt angezeigt wird.

Beobachtungen / Log / Umgebung

Dieser Test wird in Android Studio ausgeführt. Als Log dient Logcat und der Device Monitor sowie der Debugger.

Besonderheiten

Dieser Test wird von keinem Skript ausgeführt, sondern manuell von einem Tester.

Abhängigkeiten

<T202>, <T203>

5.3.5 Testfall $\langle T202 \rangle$ - Processing Activity

Ziel

Überprüfung ob die Bildbearbeitung korrekt arbeitet.

Objekte/Methoden/Funktionen

Processing Activity, ImageProcessingTask

Pass/Fail Kriterien

Pass: Bei Eingabe eines Bildes wird ein korrektes, von der Eingabe verschiedenes Bild zurückgegeben.

Fail: Jedes Ergebnis was nicht dem Pass - Kriterium entspricht.

Vorbedingung

Die Processing Activity muss ein Bild von der Kamera - Komponente erhalten haben.

Einzelschritte

Der ImageProcessingTask wird gestartet, alles weitere läut automatisch ab.

Beobachtungen / Log / Umgebung

Der Test wird in Android Studio auf einem angeschlossenen Gerät mit Android 5.1 ausgeführt. Der gesamte Vorgang wird im Logcat dokumentiert, außerdem kann der Nutzer in der App selbst beobachten ob ein korrekt bearbeitetes Bild agezeigt wird.

Besonderheiten

Dieser Test wird nicht von einem Skript ausgeführt, sondern von einem Tester der das Ergebnis visuell überprüft.

Abhängigkeiten

Leptonica (TessTwo), <T201>

5.3.6 Testfall $\langle T203 \rangle$ - Processing Activity

Ziel

Überprüfung ob die Bildbearbeitung korrekt arbeitet, wenn kein Bild übergeben wird.

Objekte/Methoden/Funktionen

Processing Activity, ImageProcessingTask

Pass/Fail Kriterien

Pass: Wird kein Bild eingeben wird eine Fehlermeldung gezeigt und ins Logcat gepostet.
Fail: Jedes Ergebnis was nicht dem Pass - Kriterium entspricht.

Vorbedingung

Die Processing Activity darf kein Bild von der Kamera - Komponente erhalten haben.

Einzelschritte

Das `ImageProcessingNoPicture`-Testskript wird in Android Studio ausgeführt.

Beobachtungen / Log / Umgebung

Der Test wird in Android Studio auf einem angeschlossenen Gerät mit Android 5.1 ausgeführt. Der gesamte Vorgang wird im Logcat dokumentiert.

Besonderheiten

Dieser Test wird nicht nur von einem Skript ausgeführt, sondern der Tester sollte das Ergebnis zudem visuell überprüfen.

Abhängigkeiten

Leptonica (TessTwo), <T201>

5.3.7 Testfall $\langle T204 \rangle$ - Processing Activity

Ziel

Überprüfung ob die Texterkennung korrekt arbeitet.

Objekte/Methoden/Funktionen

Processing Activity, OCRTask, textRecognition(Bitmap pic)

Pass/Fail Kriterien

Pass: Bei Eingabe eines Bildes wird der erkannte Text zurück gegeben.

Fail: Jedes Ergebnis was nicht dem Pass - Kriterium entspricht.

Vorbedingung

Der `OCRTask` muss ein qualitativ ausreichendes Bild vom `ImageProcessingTask` erhalten haben.

Einzelschritte

Der `OCRTask` wird vom `ImageProcessingTask` gestartet, alles weitere läuft automatisch ab.

Beobachtungen / Log / Umgebung

Der Test wird in Android Studio auf einem angeschlossenen Gerät mit Android 5.1 ausgeführt. Der gesamte Vorgang wird im Logcat dokumentiert, außerdem kann der Nutzer dort auch die erkannten Wörter nachlesen.

Besonderheiten

Dieser Test wird nicht von einem Skript ausgeführt, sondern von einem Tester der das Ergebnis visuell überprüft.

Abhängigkeiten

TessTwo

5.3.8 Testfall $\langle T300 \rangle$ - ResultActivity

Ziel

Überprüfung ob ein negatives Resultat Korrekt dargestellt wird.

Objekte/Methoden/Funktionen

ResultActivity, showResult(Boolean result)

Pass/Fail Kriterien

Pass: Es wird ein weißes Kreuz auf rotem Grund angezeigt, dass die negative Rückmeldung darstellt.

Fail: Alle anderen Ereignisse.

Vorbedingung

Ein negatives Resultat wurde vom Server an die App gesendet.

Einzelschritte

Der Nutzer muss in diesem Fall nichts tun, da das Resultat automatisch angezeigt wird, soblad das Ergebniss vom Server empfangen wurde.

Beobachtungen / Log / Umgebung

Der Test wird auf einem Android fähigen Endgerät ausgeführt, als Log dienen in diesem Fall die Beobachtungen des Testers.

Besonderheiten

Dieser Test wird nicht mit Hilfe eines Skriptes durchgeführt, da es vorallem auf die optische Rückmeldung an den Nutzer ankommt.

Abhängigkeiten

<T302>, <T304>, <T305>, <T306>.

5.3.9 Testfall $\langle T301 \rangle$ - ResultActivity

Ziel

Überprüfung ob ein positives Resultat Korrekt dargestellt wird.

Objekte/Methoden/Funktionen

ResultActivity, showResult(Boolean result)

Pass/Fail Kriterien

Pass: Es wird ein weißer Hacken auf grünem Grund angezeigt, der die postive Rückmeldung darstellt.

Fail: Alle anderen Ereignisse.

Vorbedingung

Ein positives Resultat wurde vom Server an die App gesendet.

Einzelschritte

Der Nutzer muss in diesem Fall nichts tun, da das Resultat automatisch angezeigt wird, soblad das Ergebniss vom Server empfangen wurde.

Beobachtungen / Log / Umgebung

Der Test wird auf einem Android fähigen Endgerät ausgeführt, als Log dienen in diesem Fall die Beobachtungen des Testers.

Besonderheiten

Dieser Test wird nicht mit Hilfe eines Skriptes durchgeführt, da es vorallem auf die optische Rückmeldung an den Nutzer ankommt.

Abhängigkeiten

<T302>, <T304>, <T305>, <T306>.

5.3.10 Testfall $\langle T302 \rangle$ - ReviewActivity

Ziel

Überprüfen, ob die korrekte Anzahl an Kursfeldern erzeugt wird, wenn sie kleiner als 18 ist.

Objekte/Methoden/Funktionen

ReviewActivity, creatingTextboxes(List<Course> list)

Pass/Fail Kriterien

Pass: Es wurde die korrekt Anzahl an Textfeldern erzeugt.(Anzahl der Kurfelder ist gleich der Anzahl der Listen Elemente durch zwei).

Fail: Falls die Pass-Bedingung nicht erfüllt wird.

Vorbedingung Es wird eine Liste mit weniger als 19 Kursen übergeben.

Einzelschritte

Das Erzeugen der Textfelder passiert automatisch, sobald die Activity aufgerufen wurde, daher sind keine weiteren Schritte notwendig.

Beobachtungen / Log / Umgebung

Der Test wird auf einem Android fähigen Endgerät ausgeführt, als Log dienen in diesem Fall die Beobachtungen des Testers.

Besonderheiten

Dieser Test wird nicht mit Hilfe eines Skriptes durchgeführt, da es vorallem auf die optische Rückmeldung an den Nutzer ankommt.

Abhängigkeiten

-

5.3.11 Testfall $\langle T303 \rangle$ - ReviewActivity

Ziel

Überprüfen, ob die App korrekt auf eine Eingabe mit mehr als 18 Kursfeldern reagiert.

Objekte/Methoden/Funktionen

ReviewActivity, creatingTextboxes(List<Course> list)

Pass/Fail Kriterien

Pass: Es wird eine Fehlermeldung angezeigt, die darauf hinweist, dass das Zeugnis nicht richtig erkannt wurde.

Fail: Alle anderen Ergebnisse außer der Pass-Bedingung

Vorbedingung Es wird eine Liste mit mehr als 18 Kursen übergeben.

Einzelschritte

In Android Studio wird das `CreatingTextboxesTest`-Skript ausgeführt.

Beobachtungen / Log / Umgebung

Der Test wird in Android Studio durchgeführt. Als log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.12 Testfall $\langle T304 \rangle$ - ReviewActivity

Ziel

Überprüfen, ob die statischen Felder, wie zum beispiel Name und Vorname, korrekt gesetzt werden.

Objekte/Methoden/Funktionen

ReviewActivity, settingStaticContent(), Certificate cert

Pass/Fail Kriterien

Pass: Der Inhalt der statischen Textfelder wurde korrekt gesetzt und die Werte aus dem Zeugnis werden somit angezeigt.

Vorbedingung

In der Review ist ein korrektes Certificate Objekt vorhanden.

Einzelschritte Der Inhalt der Textfelder wird automatisch gesetzt sobald die ReviewActivity angezeigt wird, daher sind keine weiteren Schritte nötig.

Beobachtungen / Log / Umgebung Der Test wird auf einem Android fähigen Endgerät ausgeführt, als Log dienen in diesem Fall die Beobachtungen des Testers.

Besonderheiten

Dieser Test wird nicht mit Hilfe eines Skriptes durchgeführt, da es vorallem auf die optische Rückmeldung an den Nutzer ankommt.

Abhängigkeiten -

5.3.13 Testfall ⟨$T305$⟩ - ReviewActivity

Ziel Überprüfen, ob der Inhalt des Zeugnisses korrekt geändert wird, nachdem der Nutzer den Inhalt der Textfelder verändert hat.

Objekte/Methoden/Funktionen ReviewActivity, updateCertificate(Context context), Certificate cert

Pass/Fail Kriterien Pass: Die Werte im in der ReviewActivity vorhandenen Zeugniss wurden entsprechend der Nutzereingaben geändert.
Fail: Alle Ergebnisse die außerhalb des Pass-Kriterium auftreten können.

Vorbedingung Der Inhalt der Reviewfelder wurde vom Nutzer geändert und kein Feld ist leer.

Einzelschritte
Nachdem der Nutzer den Inhalt der Reviewfelder verändert hat, wird der "VerfiyButton gedrückt. Danach wird automatisch die `updateCertificate` Methode aufgerufen.

Beobachtungen / Log / Umgebung
Der Test wird auf einem Android fähigen Endgerät ausgeführt, ob die Atribute korrekt geändert wurden lässt sich aus dem Logcat auslesen.

Besonderheiten
Dieser Test wird nicht mit Hilfe eines Skriptes durchgeführt, da es vorallem auf die optische Rückmeldung an den Nutzer ankommt.

Abhängigkeiten
<T302>, <T304>

5.3.14 Testfall $\langle T306 \rangle$ - ReviewActivity

Ziel

Überprüfen, ob eine Zeile aus Kurs- und Notenfeld korrekt ignoriert wird wenn sie leer ist.

Objekte/Methoden/Funktionen

ReviewActivity, updateCertificate(Context context), Certificate cert

Pass/Fail Kriterien

Pass: Die leergelassene Zeile wird für die Hash-Berechnung ignoriert.

Fail: Die leergelassene Zeile wird bei der Hash-Berechnung nicht ignoriert.

Vorbedingung

Es existiert ein korrektes Zeugnis in der `ReviewActivity`.

Einzelschritte

Der Nutzer löscht den Inhalt eines Kurs- und des dazugehörigen Notenfeldes. Danach drückt er auf Verfiy.

Beobachtungen / Log / Umgebung

Ob der korrekte Hash erzeugt wurde lässt sich im Logcat auslesen. Als Tesumgebung wird Android Studio verwendet.

Besonderheiten

Dieser Test wird nicht mit Hilfe eines Skriptes durchgeführt, da es vorallem auf die optische Rückmeldung an den Nutzer ankommt.

Abhängigkeiten

<T302>, <T304>

5.3.15 Testfall $\langle T307 \rangle$ - ReviewActivity

Ziel

Überprüfen, ob das manuelle Hinzufügen von Kurs- bzw. Notenfeldern korrekt funktioniert und nicht mehr als 18 Kursfelder hinzugefügt werden können.

Objekte/Methoden/Funktionen

ReviewActivity, userAddsNewCoursRow(View v)

Pass/Fail Kriterien

Pass: Es werden solange neue Kurs- bzw Notenfelder erzeugt, bis 18 Zeilen vorhanden sind. Danach wird eine Fehlermeldung angezeigt.

Fail: alle anderen Ergebnisse.

Vorbedingung

In der `Review Activity` wird zumindest ein Kurs- bzw. Notenfeld angezeigt.

Einzelschritte

Es wird solange auf den "PlusButton gedrückt, bis 18 Zeilen vorhanden sind. Danach wird nur ein weiteres mal auf den Button gedrückt.

Beobachtungen / Log / Umgebung

Der Test wird auf einem Android fähigen Endgerät ausgeführt, als Log dienen in diesem Fall die Beobachtungen des Testers.

Besonderheiten

Dieser Test wird nicht mit Hilfe eines Skriptes durchgeführt, da es vorallem auf die optische Rückmeldung an den Nutzer ankommt.

Abhängigkeiten

<T302>, <T304>

5.3.16 Testfall $\langle T400 \rangle$ - Communication

Ziel

Überprüfung, ob die Hashfunktionen einen korrekten Hash berechnet.

Objekte/Methoden/Funktionen

Communication, getHash(String input), bytesToHex(byte[] bytes)

Pass/Fail Kriterien

Pass: Der korrekte Hash wurde berechnet.

Fail: Es wurde kein, oder der falsche Hash berechnet.

Vorbedingung

Die Hashfunktion wird mit einem nichtleeren String aufgerufen.

Einzelschritte

Die Hashfunktion wird aufgerufen. Danach berechnet diese den Hash und ruft automatisch die `bytesToHex` Methode auf, um den Hash als Hex-Wert auszugeben.

Beobachtungen / Log / Umgebung

Der Test wird unter Android Studio durchgeführt und die Logs sind in Logcat vorhanden.

Besonderheiten

-

Abhängigkeiten

-

5.3.17 Testfall $\langle T401 \rangle$ - Communication

Ziel

Überprüfen, ob eine Verbindung zum Server aufgebaut wird

Objekte/Methoden/Funktionen

Communication, Settings, createConnection(String secureID, String hash, Context context).

Pass/Fail Kriterien

Pass: Es wird ein HttpsConnection Objekt zurückgegeben, das die richtige Serververbindung repräsentiert.

Fail: Alle anderen möglichen Ergebnisse.

Vorbedingung

In der `Settings`-Klasse ist die richtige Server-Url defeniert.

Einzelschritte

Es wird das Testskript `CreateConnectionTest` in Android Studio aufgerufen.

Beobachtungen / Log / Umgebung

Alle relevanten Log Informationen sind dem Logcat zu entnehmen. Getestet wird in Android Studio.

Besonderheiten

-

Abhängigkeiten

<T400>.

5.3.18 Testfall ⟨$T402$⟩ - Communication

Ziel

Überprüfen, ob doe HTTPS-Verbindung mit dem richtigen Zertifikat initialisiert wurde.

Objekte/Methoden/Funktionen

Communication, Settings, createConnection(String secureID, String hash, Context context).

Pass/Fail Kriterien

Pass: Die App Verbindet sich ohne Fehler zum Server.

Fail: Es tritt ein Fehler bei der Verbindung auf.

Vorbedingung

In der `Settings`-Klasse ist die richtige Server-Url defeniert.

Einzelschritte

Die Methode wird aufgerufen.

Beobachtungen / Log / Umgebung

Der Test wird auf einem Android fähigen Endgerät ausgeführt, als Log dienen in diesem Fall die Beobachtungen des Testers.

Besonderheiten

Dieser Test wird nicht mit Hilfe eines Skriptes durchgeführt, da es vorallem auf die optische Rückmeldung an den Nutzer ankommt.

Abhängigkeiten

<T400>

5.3.19 Testfall $\langle T403 \rangle$ - Communication

Ziel

Überprüfen, ob die Antwort vom Server korrekt verarbeitet wird.

Objekte/Methoden/Funktionen

Communication, getResultFromServer(HttpsURLConnection conn)

Pass/Fail Kriterien

Pass: Die Antwort des Servers wird als String von der Methode zurückgegeben.

Fail: Alle anderen Ereignisse.

Vorbedingung

Es besteht eine funktionierende Verbindung zum Server.

Einzelschritte

Der Tester startet die App und validiert ein Zeugniss. An dem Punkt an dem er in der `ReviewActivity` den Verfiy-Button drückt wird unter anderem die `getResultFromServer(HttpsURLConnection conn)`-Methode aufgerufen. Ob das Ergebnis korrekt ist, kann der Tester nun dem Logcat entnehmen.

Beobachtungen / Log / Umgebung

Logcat, Benutzerbeobachtungen.

Besonderheiten

Dieser Test wird nicht mit einem Skript ausgeführt, sondern unter realen Bedinungen von einem Tester.

Abhängigkeiten

<T400>, <T401>, <T402>

5.3.20 Testfall $\langle T404 \rangle$ - Communication

Ziel

Überprüfen ob die Fehlermeldung bei einer nicht vorhandenen Internetverbindung korrekt angezeigt wird.

Objekte/Methoden/Funktionen

Communication, getResultFromServer(HttpsURLConnection conn)

Pass/Fail Kriterien

Pass: Es wird eine Fehlermeldung angezeigt, die darauf hinweist, dass es ein Verbindungsproblem gibt.

Fail: Alle anderen möglichen Ereignisse.

Vorbedingung

Das Testgerät darf über keine Internetverbindung verfügen.

Einzelschritte

Der Tester validiert ein Zeugnis und beobachtet was nach dem Druck auf den Verfiy-Button in der `ReviewActivity` passiert.

Beobachtungen / Log / Umgebung

Der Test wird auf einem Android fähigen Endgerät ausgeführt, als Log dienen in diesem Fall die Beobachtungen des Testers.

Besonderheiten

Dieser Test wird nicht mit Hilfe eines Skriptes durchgeführt, da es vorallem auf die optische Rückmeldung an den Nutzer ankommt.

Abhängigkeiten

<T400>, <T401>, <T402>

5.3.21 Testfall $\langle T405 \rangle$ - Communication

Ziel

Überprüfen, ob eine gültige URL erzeugt wird.

Objekte/Methoden/Funktionen

Communication, createRequestURL(String serverIP, String secureID, String hash).

Pass/Fail Kriterien

Pass: die zurückgegebene URL hat die Form: ßerverIP/secureID/hash"

Fail: Es wird garkeine oder eine falsche URL zurückgegeben.

Vorbedingung

-

Einzelschritte

Das Testskript `CreateRequestURLTest` wird in Android Studio ausgeführt.

Beobachtungen / Log / Umgebung

Der Test wird in Android Studio durchgeführt, als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

<T400>

5.3.22 Testfall $\langle T406 \rangle$ - Communication

Ziel

Überprüfen, ob die Antwort des Servers korrekt in einen boolean übersetzt wird.

Objekte/Methoden/Funktionen

Communication, parseAnswer(String input)

Pass/Fail Kriterien

Pass: es wird false zurückgegeben

Fail: es wird true zurückgegeben oder es tritt ein Fehler auf

Vorbedingung

-

Einzelschritte

`ParseAnswerTestEmpty` wird in Android Studio aufgerufen.

Beobachtungen / Log / Umgebung

Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.23 Testfall $\langle T407 \rangle$ - Communication

Ziel

Überprüfen, ob die Antwort des Servers korrekt in einen boolean übersetzt wird.

Objekte/Methoden/Funktionen

Communication, parseAnswer(String input)

Pass/Fail Kriterien

Pass: Die Methode liefert true zurück.

Fail: Es wird false zurückgegeben, oder es tritt ein Fehler auf.

Vorbedingung

-

Einzelschritte

Das Testskript `ParseAnswerTestPositive` wird in Android Studio aufgerufen.

Beobachtungen / Log / Umgebung

Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.24 Testfall $\langle T408 \rangle$ - CommunicationTask

Ziel

Überprüfen, ob die Communication in einem eigenen Thread ausgeführt wird.

Objekte/Methoden/Funktionen

CommunicationTask, ReviewActivity, Communication, goToResult(View v)

Pass/Fail Kriterien

Pass: Die Kommunikation wird in ihrem eigenen Thread ausgeführt.

Fail: Alle anderen möglichen Ergebnisse.

Vorbedingung

-

Einzelschritte

Der Tester drückt in der `ReviewActivity` auf den Verify-Button und überprüft in Android Studio im Device Monitor ob ein neuer Thread erzeugt wird, in dem die Kommunikation mit dem Server stattfindet.

Beobachtungen / Log / Umgebung

Dieser Test wird in Android Studio ausgeführt. Als Log dient Logcat und der Device Monitor.

Besonderheiten

Dieser Test wird von keinem Skript ausgeführt, sondern manuell von einem Tester. Dabei zu beachten ist, das vorher die Unit Tests der <C30> Verification-Komponenten abgeschlossen sein muss und beide Komponenten integriert wurden.

Abhängigkeiten

<C30> korrekt getestet

5.3.25 Testfall ⟨$T409$⟩ - Communication

Ziel

Überprüfen, ob ein gefälschtes Cert abgelehnt wird.

Objekte/Methoden/Funktionen

Communication, validateCertificate(String id, String content, Context context)

Pass/Fail Kriterien

Pass: Die Methode gibt false zurück.

Fail: Die Methode gibt true zurück, oder es tritt ein Fehler auf.

Vorbedingung

Es wurde ein falsches Zeugnis eingelesen und es existiert eine korrekte ServerURL. Außerdem wird eine Internet Verbindung benötigt.

Einzelschritte

Die Methode `validateCertificate` wird aufgerufen.

Beobachtungen / Log / Umgebung

Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

Dieser Test führt alle Methoden der `Communication`-Klasse zusammen und sollte daher erst ausgeührt werden, wenn alle anderen Tests dieser Klasse postiv verlaufen sind. Auch hier wird kein Skript verwendet.

Abhängigkeiten

<T400>, <T401>, <T402>, <T403>, <T405>, <T407>, <T408>

5.3.26 Testfall $\langle T410 \rangle$ - Communication

Ziel

Überprüfen, ob ein valides Zeugnis akzeptiert wird.

Objekte/Methoden/Funktionen

Communication, validateCertificate(String id, String content, Context context)

Pass/Fail Kriterien

Pass: Die Methode gibt true zurück.

Fail: Die Methode gibt false zurück, oder es tritt ein Fehler auf.

Vorbedingung

Es wurde ein valides Zeugnis eingelesen und es existiert eine korrekte ServerURL. Außerdem wird eine Internet Verbindung benötigt.

Einzelschritte

Die Mehtode `validateCertificate` wird aufgerufen.

Beobachtungen / Log / Umgebung

Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

Dieser Test führt alle Methoden der `Communication`-Klasse zusammen und sollte daher erst ausgeührt werden, wenn alle anderen Tests dieser Klasse positiv verlaufen sind.

Abhängigkeiten

<T400>, <T401>, <T402>, <T403>, <T405>, <T407>, <T408>

5.3.27 Testfall $\langle T500 \rangle$ - Certificate

Ziel

Überprüfen ob die Klassenvariablen bei leerem String ocr auch leer bleiben.

Objekte/Methoden/Funktionen

Certificate(String ocr, Context c)

Pass/Fail Kriterien

Pass: Die Klassenvariablen Bleiben leer.

Fail: Die Klasse produziert einen Fehler oder die Klassenvariablen haben Inhalt.

Vorbedingung

Der String ocr wird deklariert und leer gelassen.

Einzelschritte

Der Konstruktor Certificate wird mit dem String ocr aufgerufen.

Beobachtungen / Log / Umgebung

Alle Klassenvariablen werden ausgelesen. Dieser Test wird in Android Studio durchgeführt.

Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.28 Testfall $\langle T501 \rangle$ - Certificate

Ziel

Überprüfen ob die Klassenvariablen bei inkorrekt formatiertem String ocr lediglich inkorrekt sind.

Objekte/Methoden/Funktionen

Certificate(String ocr, Context c)

Pass/Fail Kriterien

Pass: Die Klassenvariablen haben beliebigen Inhalt.

Fail: Die Klasse produziert einen Fehler.

Vorbedingung

Der String ocr wird deklariert und mit inkorrekt formatiertem Text gefüllt.

Einzelschritte

Der Konstruktor Certificate wird mit dem String ocr aufgerufen.

Beobachtungen / Log / Umgebung

Alle Klassenvariablen werden ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.29 Testfall $\langle T502 \rangle$ - Certificate

Ziel

Überprüfen ob die Klassenvariablen bei korrekt formatiertem String ocr auch korrekt sind.

Objekte/Methoden/Funktionen

Certificate(String ocr, Context c)

Pass/Fail Kriterien

Pass: Die Klassenvariablen haben den korekten Inhalt.

Fail: Die Klasse produziert einen Fehler oder der Inhalt ist inkorrekt.

Vorbedingung

Der String ocr wird deklariert und mit korrekt formatiertem Text gefüllt.

Einzelschritte

Der Konstruktor Certificate wird mit dem String ocr aufgerufen.

Beobachtungen / Log / Umgebung

Alle Klassenvariablen werden ausgelesen. Dieser Test wird in Android Studio durchgeführt.

Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.30 Testfall $\langle T503 \rangle$ - Certificate

Ziel

Überprüfen ob der User bei leerem String lastname aufgefordert wird einen Nachnamen anzugeben.

Objekte/Methoden/Funktionen

Certificate.setLastname(String lastname, Context c)

Pass/Fail Kriterien

Pass: Der User wird durch einen Dialog aufgefordert einen Nachnamen anzugeben.

Fail: Die Klasse produziert einen Fehler oder die Klassenvariable Word lastname wird auf leeren content gesetzt.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String lastname wird leer deklariert und anschließend setLastname(String lastname, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.31 Testfall $\langle T504 \rangle$ - Certificate

Ziel

Überprüfen ob Leerzeichen vom Anfang und Ende des String lastname entfernt werden und dieser dann als content der Klassenvariable Word lastname gesetzt wird.

Objekte/Methoden/Funktionen

Certificate.setLastname(String lastname, Context c)

Pass/Fail Kriterien

Pass: Leerzeichen vom Anfang und Ende des String lastname werden entfernt und dieser dann als content der Klassenvariable Word lastname gesetzt.

Fail: Es bleiben Leerzeichen vom Anfang oder Ende des String lastname oder dieser wird nicht als content der Klassenvariable Word lastname gesetzt.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String lastname wird als Wort mit Leerzeichen am Anfang und Ende deklariert und anschließend setLastname(String lastname, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.32 Testfall $\langle T505 \rangle$ - Certificate

Ziel

Überprüfen ob der User bei Ziffern im String lastname gefragt wird ob er diese entfernen möchte.

Objekte/Methoden/Funktionen

Certificate.setLastname(String lastname, Context c)

Pass/Fail Kriterien

Pass: Der User wird gefragt wird ob er die Ziffern entfernen möchte.

Fail: Die Klasse produziert einen Fehler oder der User wird nicht gefragt wird ob er die Ziffern entfernen möchte.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String lastname wird mit Ziffern deklariert und anschließend setLastname(String lastname, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.33 Testfall $\langle T506 \rangle$ - Certificate

Ziel

Überprüfen ob der User bei leerem String date aufgefordert wird ein Datum anzugeben.

Objekte/Methoden/Funktionen

Certificate.setDate(String date, Context c)

Pass/Fail Kriterien

Pass: Der User wird durch einen Dialog aufgefordert ein Datum anzugeben.

Fail: Die Klasse produziert einen Fehler oder die Klassenvariable Word date wird auf leeren content gesetzt.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String date wird leer deklariert und anschließend setDate(String date, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.34 Testfall $\langle T507 \rangle$ - Certificate

Ziel

Überprüfen ob bei String date im Format YYYY-MM-DD der String direkt als content der Klassenvariable Word date gesetzt wird.

Objekte/Methoden/Funktionen

Certificate.setDate(String date, Context c)

Pass/Fail Kriterien

Pass: String date wird direkt als content der Klassenvariable Word date gesetzt.

Fail: Die Klasse produziert einen Fehler oder der String date wird nicht direkt als content der Klassenvariable Word date gesetzt.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String date wird im Format YYYY-MM-DD deklariert und anschließend setDate(String date, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.35 Testfall $\langle T508 \rangle$ - Certificate

Ziel

Überprüfen ob bei String date im Format DD. Month YYYY der String zunächst ins Format YYYY-MM-DD umgewandelt wird und dann als content der Klassenvariable Word date gesetzt wird.

Objekte/Methoden/Funktionen

Certificate.setDate(String date, Context c)

Pass/Fail Kriterien

Pass: Der String date wird zunächst ins Format YYYY-MM-DD umgewandelt und dann als content der Klassenvariable Word date gesetzt wird.

Fail: Die Klasse produziert einen Fehler, der String date wird nicht ins Format YYYY-MM-DD umgewandelt oder nicht als content der Klassenvariable Word date gesetzt wird.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String date wird im Format DD. Month YYYY deklariert und anschließend setDate(String date, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.36 Testfall $\langle T509 \rangle$ - Certificate

Ziel

Überprüfen ob der User bei falsch formatiertem String date aufgefordert wird ein richtig formatiertes Datum anzugeben.

Objekte/Methoden/Funktionen

Certificate.setDate(String date, Context c)

Pass/Fail Kriterien

Pass: Der User wird durch einen Dialog aufgefordert ein richtig formatiertes Datum anzugeben.

Fail: Die Klasse produziert einen Fehler oder für die Klassenvariable Word date wird ein content gesetzt.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String date wird falsch formatiert deklariert und anschließend setDate(String date, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.37 Testfall $\langle T510 \rangle$ - Certificate

Ziel

Überprüfen ob der User bei leerem String name aufgefordert wird einen Namen anzugeben.

Objekte/Methoden/Funktionen

Certificate.setName(String name, Context c)

Pass/Fail Kriterien

Pass: Der User wird durch einen Dialog aufgefordert einen Namen anzugeben.

Fail: Die Klasse produziert einen Fehler oder die Klassenvariable Word name wird auf leeren content gesetzt.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String name wird leer deklariert und anschließend setName(String name, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.38 Testfall $\langle T511 \rangle$ - Certificate

Ziel

Überprüfen ob Leerzeichen vom Anfang und Ende des String name entfernt werden und dieser dann als content der Klassenvariable Word name gesetzt wird.

Objekte/Methoden/Funktionen

Certificate.setName(String name, Context c)

Pass/Fail Kriterien

Pass: Leerzeichen vom Anfang und Ende des String name werden entfernt und dieser dann als content der Klassenvariable Word name gesetzt.

Fail: Es bleiben Leerzeichen vom Anfang oder Ende des String name oder dieser wird nicht als content der Klassenvariable Word name gesetzt.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String name wird als Wort mit Leerzeichen am Anfang und Ende deklariert und anschließend setName(String name, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.39 Testfall $\langle T512 \rangle$ - Certificate

Ziel

Überprüfen ob der User bei Ziffern im String name gefragt wird ob er diese entfernen möchte.

Objekte/Methoden/Funktionen

Certificate.setName(String name, Context c)

Pass/Fail Kriterien

Pass: Der User wird gefragt wird ob er die Ziffern entfernen möchte.

Fail: Die Klasse produziert einen Fehler oder der User wird nicht gefragt wird ob er die Ziffern entfernen möchte.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String name wird mit Ziffern deklariert und anschließend setName(String name, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.40 Testfall $\langle T513 \rangle$ - Certificate

Ziel

Überprüfen ob der User bei leerem String institution aufgefordert wird einer Institution anzugeben.

Objekte/Methoden/Funktionen

Certificate.setInstitution(String institution, Context c)

Pass/Fail Kriterien

Pass: Der User wird durch einen Dialog aufgefordert eine Institution anzugeben.

Fail: Die Klasse produziert einen Fehler oder die Klassenvariable Word institution wird auf leeren content gesetzt.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String institution wird leer deklariert und anschließend setInstitution(String institution, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.41 Testfall $\langle T514 \rangle$ - Certificate

Ziel

Überprüfen ob Leerzeichen vom Anfang und Ende des String institution entfernt werden und dieser dann als content der Klassenvariable Word institution gesetzt wird.

Objekte/Methoden/Funktionen

Certificate.setInstitution(String institution, Context c)

Pass/Fail Kriterien

Pass: Leerzeichen vom Anfang und Ende des String institution werden entfernt und dieser dann als content der Klassenvariable Word institution gesetzt.

Fail: Es bleiben Leerzeichen vom Anfang oder Ende des String institution oder dieser wird nicht als content der Klassenvariable Word institution gesetzt.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String institution wird als Wort mit Leerzeichen am Anfang und Ende deklariert und anschließend setInstitution(String institution, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.42 Testfall $\langle T515 \rangle$ - Certificate

Ziel

Überprüfen ob der User bei leerem String secureID aufgefordert wird eine SecureID anzugeben.

Objekte/Methoden/Funktionen

Certificate.setSecureID(String secureID, Context c)

Pass/Fail Kriterien

Pass: Der User wird durch einen Dialog aufgefordert eine SecureID anzugeben.

Fail: Die Klasse produziert einen Fehler oder die Klassenvariable Word secureID wird auf leeren content gesetzt.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String secureID wird leer deklariert und anschließend setSecureID(String secureID, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.43 Testfall $\langle T516 \rangle$ - Certificate

Ziel

Überprüfen ob bei String secureID im Format xxxxxxxxxxxxxxx der String direkt als content der Klassenvariable Word secureID gesetzt wird.

Objekte/Methoden/Funktionen

Certificate.setSecureID(String secureID, Context c)

Pass/Fail Kriterien

Pass: String secureID wird direkt als content der Klassenvariable Word secureID gesetzt.
Fail: Die Klasse produziert einen Fehler oder der String secureID wird nicht direkt als content der Klassenvariable Word secureID gesetzt.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String secureID wird im Format xxxxxxxxxxxxxxx deklariert und anschließend setSecureID(String secureID, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.44 Testfall $\langle T517 \rangle$ - Certificate

Ziel

Überprüfen ob bei String secureID im Format xxxx-xxxx-xxxx-xxxx der String zunächst ins Format xxxxxxxxxxxxxxxx umgewandelt wird und dann als content der Klassenvariable Word secureID gesetzt wird.

Objekte/Methoden/Funktionen

Certificate.setSecureID(String secureID, Context c)

Pass/Fail Kriterien

Pass: Der String secureID wird zunächst ins Format xxxxxxxxxxxxxxxx umgewandelt und dann als content der Klassenvariable Word secureID gesetzt wird.
Fail: Die Klasse produziert einen Fehler, der String secureID wird nicht ins Format xxxxxxxxxxxxxxxx umgewandelt oder nicht als content der Klassenvariable Word secureID gesetzt wird.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String secureID wird im Format xxxx-xxxx-xxxx-xxxx deklariert und anschließend setSecureID(String secureID, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.45 Testfall $\langle T518 \rangle$ - Certificate

Ziel

Überprüfen ob der User bei falsch formatiertem String secureID aufgefordert wird ein richtig formatiertes Datum anzugeben.

Objekte/Methoden/Funktionen

Certificate.setSecureID(String secureID, Context c)

Pass/Fail Kriterien

Pass: Der User wird durch einen Dialog aufgefordert ein richtig formatiertes Datum anzugeben.

Fail: Die Klasse produziert einen Fehler oder für die Klassenvariable Word secureID wird ein content gesetzt.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Der String secureID wird falsch formatiert deklariert und anschließend setSecureID(String secureID, Context c) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.46 Testfall $\langle T519 \rangle$ - Certificate

Ziel

Überprüfen ob der User bei leerer List<Course> courses aufgefordert wird Kurse in die Liste einzutragen.

Objekte/Methoden/Funktionen

Certificate.setCourses(List<Course> courses, Context context)

Pass/Fail Kriterien

Pass: Der User wird durch einen Dialog aufgefordert Kurse in die Liste einzutragen.

Fail: Die Klasse produziert einen Fehler oder die Klassenvariable List<Course> courses wird auf leere Liste gesetzt.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Die List<Course> courses wird deklariert und leer gelassen. Anschließend wird setCourses(List<Course> courses, Context context) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.47 Testfall $\langle T520 \rangle$ - Certificate

Ziel

Überprüfen ob bei gefüllter List<Course> courses die Liste in die Klassenvariable List<Course> courses kopiert wird.

Objekte/Methoden/Funktionen

Certificate.setCourses(List<Course> courses, Context context)

Pass/Fail Kriterien

Pass: Liste<Course> courses wird in die Klassenvariable List<Course> courses kopiert.
Fail: Liste<Course> courses wird nicht in die Klassenvariable List<Course> courses kopiert.

Vorbedingung

Eine Instanz von Certificate wird mit Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollem Inhalt erstellt.

Einzelschritte

Die List<Course> courses wird deklariert und mit sinnvolen Einträgen gefüllt. Anschließend wird setCourses(List<Course> courses, Context context) aufgerufen.

Beobachtungen / Log / Umgebung

Die Rückgabe der Methode wird ausgelesen. Dieser Test wird in Android Studio durchgeführt. Als Log dient Logcat.

Besonderheiten

-

Abhängigkeiten

-

5.3.48 Testfall $\langle T521 \rangle$ - Certificate

Ziel

Überprüfen ob bei leeren Klassenvariablen die toString() Methode trotzdem eine korrekt formatierte Ausgabe produziert.

Objekte/Methoden/Funktionen

Certificate.toString()

Pass/Fail Kriterien

Pass: Die Ausgabe ist korrekt formatiert.

Fail: Die Klasse produziert einen Fehler oder die Formatierung ist falsch.

Vorbedingung

Die Certificate Klasse wird mit dem Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit leeren Variablen aufgerufen.

Einzelschritte

Die Methode toString() wird aufgerufen.

Beobachtungen / Log / Umgebung

Die Ausgabe von toString() wird gespeichert.

Besonderheiten

-

Abhängigkeiten

-

5.3.49 Testfall $\langle T522 \rangle$ - Certificate

Ziel

Überprüfen ob bei sinnvoll gefüllten Klassenvariablen die toString() Methode eine korrekt formatierte Ausgabe produziert.

Objekte/Methoden/Funktionen

Certificate.toString()

Pass/Fail Kriterien

Pass: Die Ausgabe ist korrekt formatiert.

Fail: Die Klasse produziert einen Fehler oder die Formatierung ist falsch.

Vorbedingung

Die Certificate Klasse wird mit dem Konstruktor Certificate(Word lastname, Word name, Word date, Word institution, Word secureID, List<Course> courses) mit sinnvollen Variablen aufgerufen.

Einzelschritte

Die Methode toString() wird aufgerufen.

Beobachtungen / Log / Umgebung

Die Ausgabe von toString() wird gespeichert.

Besonderheiten

-

Abhängigkeiten

-

5.3.50 Testfall $\langle T700 \rangle$ - PDFGeneration

Ziel

Überprüfung, ob ein gültiges Zeugnis über die Webseite korrekt als PDF erstellt und heruntergeladen werden kann.

Objekte/Methoden/Funktionen

Certificate

Pass/Fail Kriterien

Pass: Das Zeugnis kann als PDF heruntergeladen und geöffnet werden.

Fail: Das Zeugnis kann nicht als PDF heruntergeladen werden oder die PDF-Datei ist beschädigt.

Vorbedingung

Der Server läuft und ein gültiges Zeugnis ist bereits angelegt worden. Der Benutzer ist mit Adminrechten eingeloggt.

Einzelschritte

Ein Zeugnis über die Webseite auswählen, dem Link auf die Bearbeitungsmaske folgen und den PDF Download-Link drücken. Anschließend die generierte PDF herunterladen und auf Mängel untersuchen.

Beobachtungen / Log / Umgebung

Da dieser Test manuell ausgeführt wird, dienen die Beobachtungen des Nutzers als Log.

Besonderheiten

Dieser Test wird von keinem Skript ausgeführt, sondern manuell von einem Tester.

Abhängigkeiten

-

5.3.51 Testfall $\langle T701 \rangle$ - CreateOwner

Ziel

Überprüfung, ob ein Zeugnisinhaber über die Webseite angelegt werden kann.

Objekte/Methoden/Funktionen

Owner

Pass/Fail Kriterien

Pass: Ein Zeugnisinhaber wurde korrekt angelegt.

Fail: Ein Zeugnisinhaber konnte nicht angelegt werden.

Vorbedingung

Der Server läuft und der Benutzer ist mit Adminrechten eingeloggt.

Einzelschritte

Über die entsprechenden Button der Webseite zu der Eingabemaske der Owner navigieren. Name, Nachname und Matrikelnummer eingeben und dann über den Save-Button speichern.

Beobachtungen / Log / Umgebung

Da dieser Test manuell ausgeführt wird, dienen die Beobachtungen des Nutzers als Log.

Besonderheiten

Dieser Test wird von keinem Skript ausgeführt, sondern manuell von einem Tester.

Abhängigkeiten

-

5.3.52 Testfall $\langle T702 \rangle$ - CreateInstitution

Ziel

Überprüfung, ob ein Institut über die Webseite angelegt werden kann.

Objekte/Methoden/Funktionen

Institution

Pass/Fail Kriterien

Pass: Ein Institut wurde korrekt angelegt.

Fail: Ein Institut konnte nicht angelegt werden.

Vorbedingung

Der Server läuft und der Benutzer ist mit Adminrechten eingeloggt.

Einzelschritte

Über die entsprechenden Button der Webseite zu der Eingabemaske der Institut navigieren.

Name und Logo eingeben und dann über den Save-Button speichern.

Beobachtungen / Log / Umgebung

Da dieser Test manuell ausgeführt wird, dienen die Beobachtungen des Nutzers als Log.

Besonderheiten

Dieser Test wird von keinem Skript ausgeführt, sondern manuell von einem Tester.

Abhängigkeiten

-

5.3.53 Testfall $\langle T703 \rangle$ - CreateSubject

Ziel

Überprüfung, ob ein Zeugnisfach über die Webseite angelegt werden kann.

Objekte/Methoden/Funktionen

Subject

Pass/Fail Kriterien

Pass: Ein Zeugnisfach wurde korrekt angelegt.

Fail: Ein Zeugnisfach konnte nicht angelegt werden.

Vorbedingung

Der Server läuft und der Benutzer ist mit Adminrechten eingeloggt.

Einzelschritte

Über die entsprechenden Button der Webseite zu der Eingabemaske der Subjects navigieren. Name eingeben und dann über den Save-Button speichern.

Beobachtungen / Log / Umgebung

Da dieser Test manuell ausgeführt wird, dienen die Beobachtungen des Nutzers als Log.

Besonderheiten

Dieser Test wird von keinem Skript ausgeführt, sondern manuell von einem Tester.

Abhängigkeiten

-

5.3.54 Testfall ⟨$T704$⟩ - CreateCertificate

Ziel

Überprüfung, ob ein Zeugnis über die Webseite angelegt werden kann.

Objekte/Methoden/Funktionen

Certificate, Owner, Subject, Grade, Institution

Pass/Fail Kriterien

Pass: Ein Zeugnis wurde korrekt angelegt.

Fail: Ein Zeugnis konnte nicht angelegt werden.

Vorbedingung

Der Server läuft und der Benutzer ist mit Adminrechten eingeloggt. Zeugnisinhaber, Institute und Zeugnisfächer sind bereits angelegt.

Einzelschritte

Über die entsprechenden Button der Webseite zu der Eingabemaske der Certificates navigieren. Zeugnisinhaber, Institut und die Zeugnisfächer inklusive Noten eintragen und dann über den Save-Button speichern.

Beobachtungen / Log / Umgebung

Da dieser Test manuell ausgeführt wird, dienen die Beobachtungen des Nutzers als Log.

Besonderheiten

Dieser Test wird von keinem Skript ausgeführt, sondern manuell von einem Tester. Da Zeugnisse erst angelegt werden können, wenn Zeugnisinhaber, Zeugnisfächer und Institute existieren, sollten alle dazugehörigen Tests zuerst ausgeführt werden.

Abhängigkeiten

<T701>, <T702>, <T703>

5.3.55 Testfall $\langle T705 \rangle$ - ChangeData

Ziel

Überprüfung, ob angelegte Daten sich ändern lassen.

Objekte/Methoden/Funktionen

Certificate,Institution, Owner, Subject, Grade

Pass/Fail Kriterien

Pass: Die Daten lassen sich verändern und wieder speichern.

Fail: Die Daten lassen sich nicht verändern.

Vorbedingung

Der Server läuft und der Benutzer ist mit Adminrechten eingeloggt.Zeugnisse mit Zeugnisinhaber, Institute und Zeugnisfächer sind bereits angelegt.

Einzelschritte

Über den entsprechenden Button der Webseite zu den entsprechenden Daten der Eingabemaske navigieren.Schließlich die gesuchten Daten verändern und dann über den Save-Button speichern.

Beobachtungen / Log / Umgebung

Da dieser Test manuell ausgeführt wird, dienen die Beobachtungen des Nutzers als Log.

Besonderheiten

Dieser Test wird von keinem Skript ausgeführt, sondern manuell von einem Tester.

Abhängigkeiten

<T701> CreateOwner

<T702> CreateInstitution

<T703> CreateSubject

<T704> CreateCertificate

5.3.56 Testfall ⟨$T706$⟩ - DeleteData

Ziel

Überprüfung, ob angelegte Daten sich löschen lassen.

Objekte/Methoden/Funktionen

Certificate,Institution, Owner, Subject, Grade

Pass/Fail Kriterien

Pass: Die Daten lassen sich löschen.

Fail: Die Daten lassen sich nicht löschen.

Vorbedingung

Der Server läuft und der Benutzer ist mit Adminrechten eingeloggt. Zeugnisse mit Zeugnisinhaber, Institute und Zeugnisfächer sind bereits angelegt.

Einzelschritte

Über den entsprechenden Button der Webseite zu den entsprechenden Daten der Eingabemaske navigieren. Schließlich die gesuchten Daten löschen.

Beobachtungen / Log / Umgebung

Da dieser Test manuell ausgeführt wird, dienen die Beobachtungen des Nutzers als Log.

Besonderheiten

Dieser Test wird von keinem Skript ausgeführt, sondern manuell von einem Tester.

Abhängigkeiten

<T701> CreateOwner

<T702> CreateInstitution

<T703> CreateSubject

<T704> CreateCertificate

5.3.57 Testfall $\langle T800 \rangle$ - certificates.tests.ValidationTestCase.test_good_certificate _verification()

Ziel

Überprüfen, ob ein gültigiges Zeugnis als ein solches erkannt wird.

Objekte/Methoden/Funktionen

models.Certificate, views.verify(), django.http.HTTPResponse, logging.logger, json.JSONEncoder, django.contrib.auth.models.AnonymousUser, django.test.Client, django.test.TestCase, django.test.RequestFactory

Pass/Fail Kriterien

Pass: Das gültige Zeugnis wird korrekt erkannt.

Fail: Das gültige Zeugnis wird nicht erkannt.

Vorbedingung

Eine Kommandozeile ist geöffnet

Einzelschritte

`## ./manage.py test certificates.tests.ValidationTestCase.testtextunderscoregoodtextunderscorecertificatetextunderscoreverification`

Beobachtungen / Log / Umgebung

Ausgaben auf der Kommandozeile zeigen die erkannten Daten, verifications.log zeigt das Ergebnis der Überprüfung

Besonderheiten

-

Abhängigkeiten

-

5.3.58 Testfall ⟨T801⟩ - certificates.tests.ValidationTestCase.test_bad_certificate_falsify()

Ziel

Überprüfen, ob ein ungültigiges Zeugnis als ein solches erkannt wird.

Objekte/Methoden/Funktionen

models.Certificate, views.verify(), django.http.HTTPResponse, logging.logger, json.JSONEncoder, django.contrib.auth.models.AnonymousUser, django.test.Client, django.test.TestCase, django.test.RequestFactory

Pass/Fail Kriterien

Pass: Das ungültige Zeugnis wird korrekt als ungültig erkannt.

Fail: Das ungültige Zeugnis wird nicht erkannt.

Vorbedingung

Eine Kommandozeile ist geöffnet

Einzelschritte

`##./manage.py test certificates.tests.ValidationTestCase.testtextunderscorebadtextunderscorecertificatetextunderscorefalsify`

Beobachtungen / Log / Umgebung

Ausgaben auf der Kommandozeile zeigen die erkannten Daten. verifications.log zeigt das Ergebnis der Überprüfung.

Besonderheiten

-

Abhängigkeiten

-

5.3.59 Testfall $\langle T900 \rangle$ - GenerateSecureIdAndHash

Ziel

Überprüfung, ob korrekte Secure-IDs und Hash-Werte für Zeugnisse berechnet werden.

Objekte/Methoden/Funktionen

Certificate, Grade, Owner, Subject, createCertificate, test_secureid_and_hash

Pass/Fail Kriterien

Pass: Keine doppelten Secure-IDs wurden erzeugt und die Hash-Werte bei verschiedenen Daten waren ebenfalls verschieden.

Fail: Doppelte Secure-IDs wurden erzeugt oder die Hash-Werte verschiedener Daten waren identisch.

Vorbedingung

Eine Kommandozeile ist geöffnet

Einzelschritte

Das Testskript CertificateSecureIdAndHashTest per Konsole ausführen.

```
##./manage.py test certificates.tests.CertificateSecureIdAndHashTest.test¬
_secureid_and_hash
```

Beobachtungen / Log / Umgebung

Ausgaben auf der Kommandozeile zeigen die erkannten Daten. verifications.log zeigt das Ergebnis der Überprüfung.

Besonderheiten

-

Abhängigkeiten

-

5.3.60 Testfall ⟨$T1200$⟩ - certificates.tests.RestApiPostTestCase.test_good_full_certificate

Ziel

Überprüfen, ob ein Zeugnis und alle dazugehörigen Daten angelegt werden können.

Objekte/Methoden/Funktionen

certificates.viewsets, django.test, django.http.HTTPResponse, rest_framework.test, rest_framewok.statu

Pass/Fail Kriterien

Pass: Alle Daten wurden korrekt hinzugefügt.

Fail: Eine oder mehrere Datensätze wurden nicht hinzugefügt.

Vorbedingung

Eine Kommandozeile ist geöffnet.

Einzelschritte

```
## ./manage.py test certificates.tests.RestApiPostTestCase.testtextunders-
coregoodtextunderscorefulltextunderscorecertificate
```

Beobachtungen / Log / Umgebung

Ausgaben auf der Kommandozeile zeigen `OK`

Besonderheiten

-

Abhängigkeiten

-

5.3.61 Testfall ⟨$T1201$⟩ - certificates.tests.RestApiPostTestCase.test_bad_full_certificate

Ziel

Überprüfen, ob ein Zeugnis und alle dazugehörigen Daten nicht angelegt werden können, wenn sie nicht gültig sind.

Objekte/Methoden/Funktionen

certificates.viewsets, django.test, django.http.HTTPResponse, rest_framework.test, rest_framewok.statu

Pass/Fail Kriterien

Pass: Alle Daten wurden korrekt nicht hinzugefügt.

Fail: Ein oder mehrere Datensätze wurden hinzugefügt.

Vorbedingung

Eine Kommandozeile ist geöffnet.

Einzelschritte

```
## ./manage.py test certificates.tests.RestApiPostTestCase.testtextunders-
corebadtextunderscorefulltextunderscorecertificate
```

Beobachtungen / Log / Umgebung

Ausgaben auf der Kommandozeile zeigen `OK`

Besonderheiten

-

Abhängigkeiten

-

5.3.62 Testfall ⟨$T1202$⟩ - certificates.tests.RestApiGetTestCase.test_good_get

Ziel

Überprüfen, ob die Zeugnisdaten über die API verfügbar sind.

Objekte/Methoden/Funktionen

certificates.viewsets, django.test, django.http.HTTPResponse, rest_framework.test, rest_framewok.statu

Pass/Fail Kriterien

Pass: Alle Daten sind über die API einsehbar, mit Ausnahme der Hash-Werte.

Fail: Ein oder mehrere Datensätze sind über die API nicht erreichbar, ausser die Hash-Werte.

Vorbedingung

Eine Kommandozeile ist geöffnet.

Einzelschritte

`## ./manage.py test certificates.tests.RestApiGetTestCase.testtextundersco-regoodtextunderscoreget`

Beobachtungen / Log / Umgebung

Ausgaben auf der Kommandozeile zeigen `OK`

Besonderheiten

-

Abhängigkeiten

<T1200>, <T1201>

5.3.63 Testfall ⟨$T1203$⟩ - certificates.tests.RestApiGetTestCase.test_bad_get

Ziel

Überprüfen, ob nur Zeugnisdaten von Zeugnissen, die Existieren über die API verfügbar sind und bei falscher Anfrage keine Daten ausgeliefert werden.

Objekte/Methoden/Funktionen

certificates.viewsets, django.test, django.http.HTTPResponse, rest_framework.test, rest_framewok.statu

Pass/Fail Kriterien

Pass: Alle Daten sind über die API einsehbar, mit Ausnahme der Hash-Werte.

Fail: Ein oder mehrere Datensätze sind über die API nicht erreichbar, ausser die Hash-Werte.

Vorbedingung

Eine Kommandozeile ist geöffnet.

Einzelschritte

`## ./manage.py test certificates.tests.RestApiGetTestCase.testtextundersco-rebadtextunderscoreget`

Beobachtungen / Log / Umgebung

Ausgaben auf der Kommandozeile zeigen `OK`

Besonderheiten

-

Abhängigkeiten

<T1200>, <T1201>

5.3.64 Testfall ⟨$T1204$⟩ - certificates.tests.RestApiDeleteTestCase.test_good_delete

Ziel

Überprüfen, ob vorhandene Daten gelöscht werden können.

Objekte/Methoden/Funktionen

certificates.viewsets, django.test, django.http.HTTPResponse, rest_framework.test, rest_framewok.statu

Pass/Fail Kriterien

Pass: Alle Daten sind über die API löschbar, mit Ausnahme von Zeugnisfächern, welche als gelöscht markiert werden sollen.

Fail: Ein oder mehrere Datensätze können nicht über die API gelöscht werden.

Vorbedingung

Eine Kommandozeile ist geöffnet.

Einzelschritte

`## ./manage.py test certificates.tests.RestApiDeleteTestCase.testtextunderscoregoodtextunderscoredelete`

Beobachtungen / Log / Umgebung

Ausgaben auf der Kommandozeile zeigen `OK`

Besonderheiten

-

Abhängigkeiten

<T1200>, <T1201>

5.3.65 Testfall ⟨$T1205$⟩ - certificates.tests.RestApiDeleteTestCase.test_bad_delete

Ziel

Überprüfen, ob nicht vorhandene Daten gelöscht werden können beziehungsweise vorhandene Daten nur unter richtigem Bezeichner gelöscht werden.

Objekte/Methoden/Funktionen

certificates.viewsets, django.test, django.http.HTTPResponse, rest_framework.test, rest_framewok.statu

Pass/Fail Kriterien

Pass: Über nicht vorhandene Bezeichner sind keine Daten löschbar.

Fail: Über einen oder mehrere nicht vorhandene Bezeichner sind Daten löschbar.

Vorbedingung

Eine Kommandozeile ist geöffnet.

Einzelschritte

`## ./manage.py test certificates.tests.RestApiDeleteTestCase.testtextunderscorebadtextunderscoredelete`

Beobachtungen / Log / Umgebung

Ausgaben auf der Kommandozeile zeigen `OK`

Besonderheiten

-

Abhängigkeiten

<T1200>, <T1201>

5.3.66 Testfall ⟨$T1206$⟩ - certificates.tests.RestApiPutTestCase.test_good_update

Ziel

Überprüfen, ob vorhandene Daten aktualisiert und ergänzt werden können.

Objekte/Methoden/Funktionen

certificates.viewsets, django.test, django.http.HTTPResponse, rest_framework.test, rest_framewok.statu

Pass/Fail Kriterien

Pass: Alle Daten sind aktualisierbar und ergänzbar.

Fail: Einzelne Daten sind nicht aktualisierbar oder ergänzbar.

Vorbedingung

Eine Kommandozeile ist geöffnet.

Einzelschritte

`## ./manage.py test certificates.tests.RestApiPutTestCase.testtextundersco-regoodtextunderscoreupdate`

Beobachtungen / Log / Umgebung

Ausgaben auf der Kommandozeile zeigen `OK`

Besonderheiten

keine

Abhängigkeiten

<T1200>, <T1201>

5.3.67 Testfall ⟨T1207⟩ - certificates.tests.RestApiPutTestCase.test_bad_update

Ziel

Überprüfen, ob Daten nur mit korrekten Daten aktualisiert werden können.

Objekte/Methoden/Funktionen

certificates.viewsets, django.test, django.http.HTTPResponse, rest_framework.test, rest_framewok.statu

Pass/Fail Kriterien

Pass: Keine nicht korrekten Daten wurden eingetragen.

Fail: Einzelne oder mehrere nicht korrekte Daten wurden eingetragen.

Vorbedingung

Eine Kommandozeile ist geöffnet.

Einzelschritte

`## ./manage.py test certificates.tests.RestApiPutTestCase.testtextundersco-rebadtextunderscoreupdate`

Beobachtungen / Log / Umgebung

Ausgaben auf der Kommandozeile zeigen `OK`

Besonderheiten

-

Abhängigkeiten

<T1200>, <T1201>

6 Glossar

Hier werden Fachbegriffe erklärt.

Bezeichnung	Beschreibung
Action Bar:	Die Action Bar befindet sich in einer Android Applikation am oberen Bildschirmrand. Sie zeigt dem Nutzer seine Lage in der Applikation und bietet verschiedene Aktionen.
OCR:	Optical Character Recognition (Zeichenerkennung) - erkennen von nicht digitalen Zeichen und deren Digitalisierung
TessBaseApi:	Ist eine Java Klasse der Schrifterkennungsbibliothek Tess Two.
Tess Two:	Android optimierter Fork der Open Source Bibliothek Tesseract.
Trainingdaten:	Datenpaket um die OCR-Bibliothek Tess Two auf die Erkennung einer Sprache zu trainieren.
Bitmap:	Rasterfeld im Speicher, das ein Bild aus einzeln ansteuerbaren Pixeln in Vertikal-/Horizontal- Koordinaten aufbaut.
URL:	Uniform Resource Locator - identifiziert und lokalisiert eine Ressource in einem Netzwerk
PDF-Datei:	Portable Document Format - ein Dateiformat welches für die plattformunabhängige Anzeige von Dokumenten benutzt wird
HASH:	Streuwertfunktion, Abbildung beliebiger Daten auf einen Wertebereich fester Größe.
HTTP:	Hypertext Transfer Protocol
SSL:	Secure Sockets Layer
SecureID:	Eine für ein Zeugnis eindeutig vergebene Identifikationsnummer, die gleichzeitig aber noch keine Aussage über den Inhalt des Zeugnis tätigt.

Aktivität:	Repräsentiert die Präsentationsebene einer Android Applikation, z.B. einen Programmbildschirm den der Benutzter sieht. Eine Android Applikation kann aus mehreren Aktivitäten bestehen und sie kann während der Laufzeit zwischen mehreren Aktivitäten wechseln.
parsen:	Maschinenlesbare Daten analysieren, segmentieren und codieren.
GUI:	Graphical User Interface.
WebInterface bzw.Webschnittstelle:	als Webschnittstelle bezeichnet man eine Schnittstelle zu einem System, die über HTTP angesprochen werden kann. Dabei handelt es sich um eine grafische Benutzeroberfläche (GUI), über die ein Benutzer mit Hilfe eines Webbrowsers mit dem System interagieren kann, oder einen Webservice, durch den das System anderen Systemen Daten oder Funktionen zur Verfügung stellt.
serverIP:	ist eine Adresse in Computernetzen, die – wie das Internet – auf dem Internetprotokoll (IP) basiert. Sie wird Geräten zugewiesen, die an das Netz angebunden sind, und macht die Geräte so adressierbar und damit erreichbar.
camera2:	Die camera2 Api wurde mit Android 5 eingeführt und ersetzt die nun als deprecated gekennzeichnete camera Api. Sie dient dazu Bilder aufzunehmen.
Man-in-the-Middle-Angriff:	Grob gesagt steht dabei ein Angreifer zwischen zwei Kommunikations-Partnern und gibt beiden vor der jeweils andere zu sein. Dadurch erlangt er volle Kontrolle über alle ausgetauschten Daten.
Multithreading:	Multithreading bedeutet das verschiedene Aufgaben zeitgleich ausgeführt werden.
TCP/IP:	(Transmission Control Protocol/Internet Protocol) ist die grundlegende Kommunikationssprache (Protokoll) des Internets. Es kann außerdem als Kommunikationsprotokoll innerhalb eines privaten Netzwerks (Intranet oder Extranet) verwendet werden.
JSON:	JavaScript Object Notation.
LDAP:	Lightweight Directory Access Protocol.
Django:	ist ein in Python geschriebenes quelloffenes Web Application Framework, das einem Model-View-Controller-Schema folgt. Benannt ist es nach dem Jazz-Gitarristen Django Reinhardt.

SQL:	Structured Query Language
IEEE 829 Standard:	ist ein vom IEEE (Institute of Electrical and Electronic Engineers) veröffentlichter Standard, der einen Satz von acht Basis-Dokumenten zur Dokumentation von Softwaretests beschreibt. Der Standard beschreibt Form und Inhalt der jeweiligen Dokumente. Er schreibt jedoch nicht vor, welche der jeweiligen Dokumente zwingend verwendet werden müssen.