Projektkontext von "Lunchify”

Sie arbeiten bei einem Linzer Software-Entwicklungsunternehmen, welches die Eigenentwicklung einer Softwarelösung für die Rückvergütung von Essensausgaben ihrer Mitarbeiter in Auftrag gibt.

Die Geschäftsführung des Unternehmens möchte mit Mitte des Jahres ein System einführen, das es ihren Mitarbeitern ermöglicht, Rechnungen ihres Mittagessens einzureichen, um die Rückvergütung eines Fixbetrages pro Tag beantragen zu können. Durch die fachliche Kompetenz im Unternehmen und die Möglichkeit die Software auch kostengünstig betreiben zu können, hat man sich für eine Eigenentwicklung der Applikation „Lunchify“ entschieden. Außerdem ist es somit leichter möglich, die Schnittstelle zum Gehaltsverrechnungssystem zu implementieren, um die Rückvergütung auch automatisieret über die monatliche Gehaltsabrechnung abwickeln zu können.

In Absprache mit der Geschäftsführung wurde bereits festgelegt, dass die Entwicklung des Systems agil stattfinden soll, um möglichst früh Feedback aller beteiligten Stakeholder einfließen lassen zu können. In einem ersten Kick-off Meeting wurden bereits die minimalen Anforderungen für „Lunchify“ erhoben und sind nachfolgend angeführt. Als Benutzer der Software wurden alle Mitarbeiter:innen des Unternehmens und Mitarbeiter:innen aus der Gehaltsverrechnung identifiziert.

Realisieren Sie nachfolgende Anforderungen unter Verwendung von Java mit geeigneten Bibliotheken entweder als Desktop-Applikation oder (wenn das gesamte Team über ausreichende Kenntnisse verfügt) als Android-Applikation bzw. Web-Applikation. Die von den Anwender:innen eingegebenen bzw. bearbeiteten Daten sollen entsprechend in einer Datei oder einer Datenbank persistiert werden.

Anforderungen

Es gibt zwei Personengruppen, für die Lunchify entwickelt wird:

1. **Benutzer**: Mitarbeiter:innen des Unternehmen, die die Applikation für das Einreichen von Rechnungen verwenden.
2. **Administratoren**: Mitarbeiter:innen des Unternehmen, die die Applikation für das Abrechnen der Rückvergütung benötigen, sowie einen Gesamtüberblick über alle eingereichten Rechnungen haben.

Benutzerauthentifizierung und -autorisierung

- Benutzer und Administratoren müssen sich mit ihrer E-Mail-Adresse und Password anmelden.
- Nur authentifizierte Benutzer können Rückerstattungsanträge einreichen.
- Nur authentifizierte Administratoren können den Gesamtüberblick einsehen und die Daten für die Rückvergütung exportieren.

Rechnungs-Upload und Klassifizierung

- Benutzer können pro Tag ein Bild oder PDF ihrer Rechnung hochladen.
- Die Anwendung sollte gängige Bildformate (JPEG, PNG, PDF) unterstützen.
- Benutzer können selbst bestimmen, ob die Rechnung von einem Restaurant oder Supermarkt stammt und den Rechnungsbetrag eingeben.
- Die Anwendung kann die Rechnung automatisch klassifizieren und erkennen, ob sie von einem Restaurant oder Supermarkt stammt.

- Die Anwendung kann den Rechnungsbetrag automatisch ermitteln.
- Verwenden Sie OCR (Optical Character Recognition), um Text aus der Quittung zur Klassifizierung zu extrahieren. (Anmerkung: Kosten für eine zusätzliche OCR-Lizenz sind nicht freigeben.)

Rückerstattungsberechnung

- Wenn die Rechnung von einem Restaurant stammt und der Rechnungsbetrag 3 € überschreitet, dann erhält der Benutzer eine Rückerstattung von 3 €. Ansonsten bekommt der Benutzer den Rechnungsbetrag.
- Wenn die Rechnung von einem Supermarkt stammt und der Rechnungsbetrag 2,5 € überschreitet, erhält der Benutzer eine Rückerstattung von 2,5 €. Ansonsten bekommt der Benutzer den Rechnungsbetrag.

Übermittlungsbestätigung

- Benutzer erhalten eine Bestätigungsnachricht, nachdem sie eine Rechnung erfolgreich übermittelt haben.
- Die Bestätigung sollte den erstatteten Betrag enthalten.

Verlauf und Korrektur

- Benutzer können ihre früheren Rückerstattungsanträge und deren Status anzeigen. Diese Ansicht zeigt zusätzliche Details wie Übermittlungsdatum, Rechnungsbetrag und Klassifizierung.
- Neben der Auflistung der Rechnungen gibt es eine Übersicht, die eine grafische Verteilung von Rechnungen aus einem Restaurant und Supermarkt darstellt.
- Benutzer können eingereichte Rechnungen bis zum letzten Tag im Monat korrigieren oder löschen. Rechnungen aus dem vorherigen Monat können nicht mehr geändert werden, da sie bei der Gehaltsabrechnung berücksichtigt wurden.

Admin-Dashboard, Suchen und Filtern

- Administratoren können alle Erstattungsanträge (grafische/tabellarische) einsehen, um die Akzeptanz der Rückvergütung zu bewerten. Fragen, die zu beantworten sind:
  - Wieviel Rechnungen wurden pro Monat eingereicht?
  - Wieviel Rechnungen werden im Durchschnitt pro Benutzer im Monat eingereicht?
  - Wie ist die Verteilung zwischen Rechnungen aus einem Restaurant oder Supermarkt?
  - Wie hoch ist die gesamte Rückvergütung für das Unternehmen pro Monat und für die letzten 12 Monate?
- Administratoren können Berichte in verschiedenen Formaten (CSV, PDF) exportiert.
- Administratoren können gezielt nach einem Benutzer suchen, um eine Rechnung zu korrigieren, löschen, oder abzulehnen.

Gehaltsabrechnung

- Administratoren können für die Gehaltsabrechnung einen Export der Rechnungen für jedes Monat durchführen. Die exportierten Daten beinalten alle Benutzer und die Summe aller Rechnungen pro Monat. Die exportierten Daten sind in einem strukturierten Datenformat (JSON oder XML) gespeichert.

Konfiguration

- Administratoren können den Betrag der Rückerstatt ändern, sodass der neue Betrag für alle weiteren Rückerstattungen gültig ist.
- Administratoren können Benutzer hinzufügen und löschen.

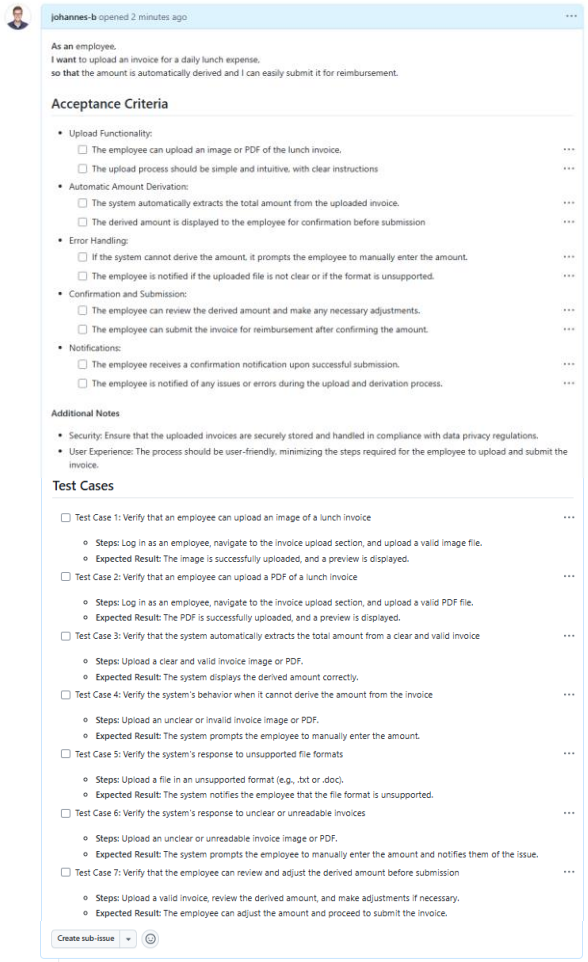
Anomalie-Erkennung

- Administratoren können Anomalien in der Verwendung der Applikation erkennen. Benutzer werden unter anderem markiert, wenn:
    - bei einer Rechnung der Rechnungsbetrag auf dem Beleg nicht mit dem eingereichten Rechnungsbetrag übereinstimmt.
    - überdurchschnittlich viele Änderungen der eingereichten Rechnungen durchführt werden.
    - *Welche weiteren Anomalien sind denkbar?*

Die Anforderungen aus der obigen Liste sind die Mindestanforderungen für die Applikation Lunchify. Bei fehlenden Informationen treffen sie bitte sinnvolle Annahmen und dokumentieren sie diese. Sie können auch gerne weitere Anforderungen einfließen lassen, um der Applikation einen individuellen Stempel aufzudrücken. Teilen sie sich die zur Verfügung stehende Zeit gut ein, sodass die Anforderungen im Budget (150 * 3 = 450 Stunden) umsetzbar sind.

Arbeiten mit User-Stories

Beispiel einer User-Story: https://github.com/jku-win-se/teaching-2025.ss.prse.template/issues/1



johannes-b opened 2 minutes ago

As an employee,
I want to upload an invoice for a daily lunch expense,
so that the amount is automatically derived and I can easily submit it for reimbursement.

Acceptance Criteria

- Upload Functionality:
  - [ ] The employee can upload an image or PDF of the lunch invoice.
  - [ ] The upload process should be simple and intuitive, with clear instructions
- Automatic Amount Derivation:
  - [ ] The system automatically extracts the total amount from the uploaded invoice.
  - [ ] The derived amount is displayed to the employee for confirmation before submission
- Error Handling:
  - [ ] If the system cannot derive the amount, it prompts the employee to manually enter the amount.
  - [ ] The employee is notified if the uploaded file is not clear or if the format is unsupported.
- Confirmation and Submission:
  - [ ] The employee can review the derived amount and make any necessary adjustments.
  - [ ] The employee can submit the invoice for reimbursement after confirming the amount.
- Notifications:
  - [ ] The employee receives a confirmation notification upon successful submission.
  - [ ] The employee is notified of any issues or errors during the upload and derivation process.

Additional Notes

- Security: Ensure that the uploaded invoices are securely stored and handled in compliance with data privacy regulations.
- User Experience: The process should be user-friendly, minimizing the steps required for the employee to upload and submit the invoice.

Test Cases

- [ ] Test Case 1: Verify that an employee can upload an image of a lunch invoice
  - **Steps:** Log in as an employee, navigate to the invoice upload section, and upload a valid image file.
  - **Expected Result:** The image is successfully uploaded, and a preview is displayed.
- [ ] Test Case 2: Verify that an employee can upload a PDF of a lunch invoice
  - **Steps:** Log in as an employee, navigate to the invoice upload section, and upload a valid PDF file.
  - **Expected Result:** The PDF is successfully uploaded, and a preview is displayed.
- [ ] Test Case 3: Verify that the system automatically extracts the total amount from a clear and valid invoice
  - **Steps:** Upload a clear and valid invoice image or PDF.
  - **Expected Result:** The system displays the derived amount correctly.
- [ ] Test Case 4: Verify the system's behavior when it cannot derive the amount from the invoice
  - **Steps:** Upload an unclear or invalid invoice image or PDF.
  - **Expected Result:** The system prompts the employee to manually enter the amount.
- [ ] Test Case 5: Verify the system's response to unsupported file formats
  - **Steps:** Upload a file in an unsupported format (e.g., .txt or .doc).
  - **Expected Result:** The system notifies the employee that the file format is unsupported.
- [ ] Test Case 6: Verify the system's response to unclear or unreadable invoices
  - **Steps:** Upload an unclear or unreadable invoice image or PDF.
  - **Expected Result:** The system prompts the employee to manually enter the amount and notifies them of the issue.
- [ ] Test Case 7: Verify that the employee can review and adjust the derived amount before submission
  - **Steps:** Upload a valid invoice, review the derived amount, and make adjustments if necessary.
  - **Expected Result:** The employee can adjust the amount and proceed to submit the invoice.

Create sub-issue

Jede User-Story hat:

- Akzeptanzkriterien,
- Testfälle (wenn sinnvoll),
- eine zugeordnete Milestone,
- und genau eine(n) Verantwortliche(n) – wird in GitHub als *Assignee* bezeichnet.

- Eine User-Story hat einen knappen, aber selbsterklärenden Titel. z.B.: *Upload Daily Lunch Invoice and Automatic Amount Derivation*

- Eine User-Story wird immer aus der Sichtweise des entsprechenden Endanwenders geschrieben. z.B.: ***As an*** *employee,*
  I want *to upload an invoice for a daily lunch expense,*
  so that *the amount is automatically derived, and I can easily submit it for reimbursement.*

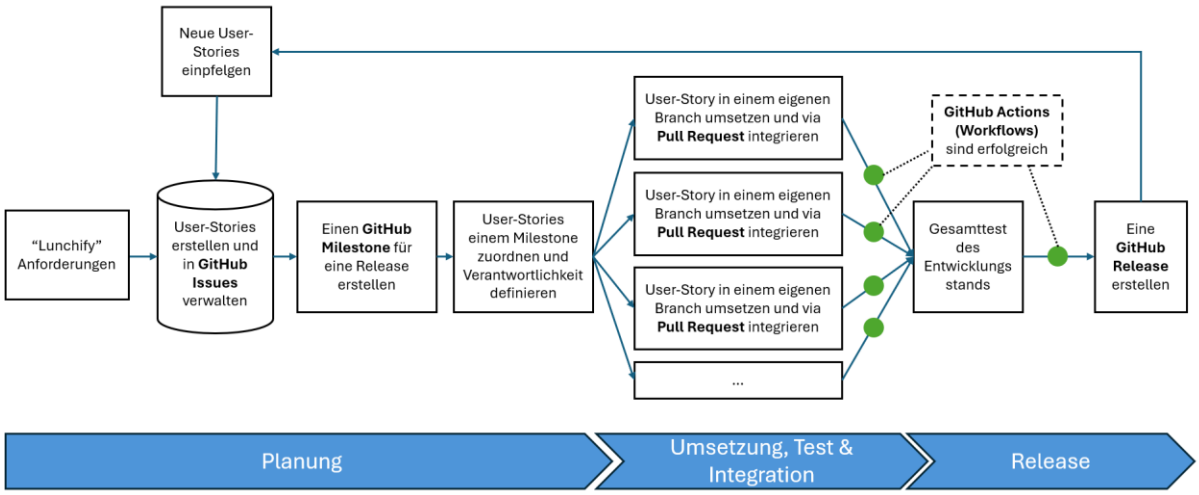
- Eine User-Story hat Akzeptanzkriterien, die erfüllt sein müssen, um die User-Story abschließen zu können.
- Eine User-Story definiert Testfälle, um die Implementierung der Akzeptanzkriterien valideren zu können.
- Für eine User-Story gilt folgende **verpflichtende Definition of Done**[1]:
  - Die User-Story ist funktional vollständig umgesetzt, d.h. alle Akzeptanzkriterien der umgesetzten User-Stories sind implementiert und getestet.

[1] Diese Anforderungen müssen erfüllt sein, bevor Sie in GitHub eine User Story als abgeschlossen markieren!

- Der Entwurf ist objekt-orientiert und Modell, View und Controller sind sinnvoll getrennt. Achten Sie in diesem Zusammenhang vor allem darauf, die Controller-Klassen klein zu halten.
- Unit-Tests mit hoher Branch-Coverage sind vorhanden
- Eine API-Dokumentation ist vorhanden
- Die technische Schuld der Implementierung ist auf Basis der SonarQube/PMD Ergebnisse sinnvoll reduziert (erst ab Release 2).

Planungs- und Umsetzungsprozess

Jedes Release soll in folgender Art und Weise geplant und realisiert werden:



- Verwenden sie **GitHub Issues**, um den Backlog an User-Stories zu verwalten.
- Erstellen sie pro Release einen **GitHub Milestone**, um die User-Stories zu bestimmen, die für die Release geplant sind.
- (optional) Verwenden sie **GitHub Projects**, um den groben Zustand der Umsetzung darzustellen und Verantwortlichkeiten zu verwalten.
- Die Verwendung von **Pull Requests** pro User Story ist verpflichtend. Details diesbezüglich folgen in der LVA.
- Eine User Story darf nur dann geschlossen werden / Ein Pull Request darf nur "*gemerged*" werden, wenn die **Definition of Done** erfüllt ist (s.o.)
- Eine User Story darf nur dann geschlossen werden / Ein Pull Request darf nur "*gemerged*" werden, wenn die **Workflows erfolgreich** sind.
- Stellen Sie dabei auch sicher (Gesamttest), dass die Akzeptanzkriterien aller umgesetzten User Stories erfüllt sind.
- Nach dem Abschluss aller User-Stories die dem Milestone zugeordnet sind, ist eine **GitHub Release** mit einem **Tag** zu erstellen (z.B. v0.0.1), damit es möglich ist, sich jederzeit diesen Release-Stand anzuschauen. Details diesbezüglich folgen in der LVA.