HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS

Zeitplanung

Programmierprojekt (DarkIT)

Phishing Defender

Ein interaktives Lernspiel zur Erkennung von Phishing-E-Mails

Student: Yusef Bach

Matrikelnummer: 1821804

Studiengang: Angewandte Informatik

Fakultät: Fakultät IV - Wirtschaft und Informatik

Hochschule: Hochschule Hannover

Hannover, 31. Oktober 2025

Inhaltsverzeichnis

1 Projektübersicht **2**
1.1 Projektziele . . . 2
1.2 Projektumfang . . . 2
1.3 Technologie-Stack . . . 2
1.4 Zeitrahmen . . . 2

2 Schritte zur Erstellung des Zeitplans **3**
2.1 1. Projektziele definieren . . . 3
2.2 2. Aufgaben identifizieren . . . 3
2.3 3. Aufgaben priorisieren . . . 3
2.4 4. Ressourcen zuweisen . . . 3
2.5 5. Zeitaufwand schätzen . . . 3
2.6 6. Meilensteine festlegen . . . 3

3 Detaillierter Zeitplan **4**

4 Benötigte Ressourcen **6**
4.1 Software-Ressourcen . . . 6
4.2 Fachliche Ressourcen . . . 6
4.3 Materialien . . . 6

5 Risikomanagement **7**

6 Meilensteine und Erfolgskriterien **8**
6.1 Haupt-Meilensteine . . . 8
6.2 Erfolgskriterien . . . 8

7 Überwachung und Berichtsmethoden **9**
7.1 Fortschrittskontrolle . . . 9
7.2 Anpassungen . . . 9

8 Fazit **10**

1 Projektübersicht

1.1 Projektziele

Das Projekt **Phishing Defender** ist ein Java-basiertes Lernspiel, das Benutzer dabei unterstützt, Phishing-E-Mails zu erkennen und zu vermeiden. Das Spiel kombiniert spielerische Elemente mit pädagogischen Inhalten, um das Bewusstsein für Cyber-Sicherheit zu stärken.

1.2 Projektumfang

- **Gameplay:** Spieler analysieren E-Mails und entscheiden, ob sie Phishing sind oder legitim
- **Level-System:** 3 Level mit steigender Schwierigkeit
- **Bewertungssystem:** Punktevergabe und Sterne-System (0-3 Sterne)
- **Highscore:** Speicherung der besten Ergebnisse mit Spieler-Namen
- **Spieler-Management:** Wechsel zwischen verschiedenen Spielern
- **Audio:** Hintergrundmusik und Sound-Effekte
- **UI/UX:** Moderne, benutzerfreundliche Oberfläche mit animiertem Hintergrund

1.3 Technologie-Stack

- **Programmiersprache:** Java 17+
- **GUI-Framework:** Java Swing
- **Versionskontrolle:** Git/GitHub
- **IDE:** IntelliJ IDEA

1.4 Zeitrahmen

12 Wochen (ca. 3 Monate)

2 Schritte zur Erstellung des Zeitplans

2.1 1. Projektziele definieren

Klare Ziele und Anforderungen des Projekts wurden festgelegt: Ein interaktives Lernspiel zur Phishing-Erkennung mit Highscore-System und mehreren Schwierigkeitsstufen.

2.2 2. Aufgaben identifizieren

Alle notwendigen Aufgaben wurden in folgende Phasen unterteilt:

- Projektinitiierung und Planung
- Anforderungsanalyse
- Design (Use-Cases, Klassenmodell, Architektur)
- Implementierung (3 Phasen)
- Testing und Optimierung
- Dokumentation
- Projektabschluss

2.3 3. Aufgaben priorisieren

Die Aufgaben wurden nach Abhängigkeiten geordnet:

1. Design vor Implementierung
2. Kernfunktionalität vor Features
3. Testing parallel zur Entwicklung
4. Dokumentation kontinuierlich

2.4 4. Ressourcen zuweisen

Entwickler: Yusef (Einzelprojekt)
Ressourcen: IntelliJ IDEA, Java SDK, Git, UML-Tools, Audio-Dateien

2.5 5. Zeitaufwand schätzen

Jede Phase wurde auf 1 Woche geschätzt, mit Pufferzeit für unvorhergesehene Probleme.

2.6 6. Meilensteine festlegen

Wichtige Meilensteine wurden definiert (siehe Tabelle im nächsten Abschnitt).

3 Detaillierter Zeitplan

Woche	Aktivität	Aufgaben	Verantwortlich	Meilenstein
1	Projektinitiierung	• Kick-off Meeting • Projektziele festlegen • Technologie-Stack wählen (Java Swing) • Repository aufsetzen	Yusef	Projektziele definiert
2	Anforderungsanalyse	• Recherche zu Phishing-Erkennung • Sammlung echter Phishing-Beispiele • Definition der Lernziele • Festlegung der Spielmechanik	Yusef	Anforderungen dokumentiert
3	Anwendungsfälle definieren	• Use-Case: Email analysieren • Use-Case: Entscheidung treffen • Use-Case: Highscore speichern • Use-Case Diagramm erstellen	Yusef	Use-Cases dokumentiert
4	Klassenmodell entwerfen	• Klassendiagramm erstellen • Hauptklassen definieren • Attribute & Methoden festlegen • Review des Modells	Yusef	Klassenmodell erstellt
5	Architekturentwurf	• MVC-Architektur definieren • Package-Struktur festlegen • Datenbank-Design (Highscores) • Sound-Integration planen	Yusef	Architektur genehmigt
6	Implementierung Phase 1	• Grundgerüst (JFrame, Panels) • Welcome Screen implementieren • Level-Auswahl Screen • Basis-Navigation	Yusef	Grundgerüst steht
7	Implementierung Phase 2	• GamePanel mit Email-Anzeige • Buttons (Phishing/Sicher) • Logik für richtig/falsch • Punkte-System implementieren	Yusef	Gameplay funktioniert
8	Implementierung Phase 3	• Highscore-System (Speichern/Laden) • Sterne-System (0-3 Sterne) • Level-Freischaltung basierend auf Sternen • Spieler-Wechsel Funktion	Yusef	Features komplett

Woche	**Aktivität**	**Aufgaben**	**Verantwortlich**	**Meilenstein**
9	Sound & UI-Polish	• Musik-Manager implementieren • Sound-Effekte einbinden • UI-Design verbessern • Animationen hinzufügen	Yusef	Sound & UI fertig
10	Testing & Bug-Fixing	• Unit-Tests für Kernlogik • Integrationstests durchführen • Bug-Fixing und Optimierung • Performance-Tests	Yusef	System stabil
11	Dokumen-tation	• Javadoc-Kommentare schreiben • README.md erstellen • UML-Diagramme finalisieren • Benutzer-Anleitung verfassen	Yusef	Dokumentation fertig
12	Projekt-abschluss	• JAR-Datei erstellen und testen • Final Testing durchführen • Abgabe vorbereiten • Präsentation erstellen	Yusef	Projekt abgegeben

4 Benötigte Ressourcen

4.1 Software-Ressourcen

- **IntelliJ IDEA:** Entwicklungsumgebung
- **Java SDK 17+:** Programmiersprache und Runtime
- **Git/GitHub:** Versionskontrolle
- **PlantUML / Draw.io:** UML-Diagramme
- **Maven/Gradle:** Build-Management

4.2 Fachliche Ressourcen

- Kenntnisse in Java und Swing
- Verständnis von OOP-Prinzipien
- Erfahrung mit File I/O und Serialization
- Grundkenntnisse in Audio-Verarbeitung (JavaSound API)

4.3 Materialien

- Sammlung echter Phishing-Beispiele
- Audio-Dateien für Musik und Sound-Effekte
- Icons und Grafiken für UI-Elemente

5 Risikomanagement

Risiko	**Wahrscheinlich-keit**	**Gegenmaßnahme**
Technische Probleme mit Swing	Mittel	Stack Overflow, Dokumentation, AI-Hilfe nutzen
Zeitüberschreitung	Hoch	Pufferzeit einplanen, Features priorisieren (MVP-Ansatz)
Datenverlust bei Highscores	Niedrig	Backup-Strategie, regelmäßige Commits in Git
JAR-Datei funktioniert nicht	Mittel	Früh testen, getResourceAsStream() für Assets nutzen
Unzureichende Tests	Mittel	Test-Plan erstellen, Unit-Tests parallel zur Entwicklung
Komplexität der Phishing-Erkennung	Mittel	Recherche, Expertenrat einholen, einfache Kriterien definieren
Performance-Probleme bei Animationen	Niedrig	Profiling durchführen, Optimierung bei Bedarf

6 Meilensteine und Erfolgskriterien

6.1 Haupt-Meilensteine

1. **Woche 1:** Projekt initiiert und Ziele definiert
2. **Woche 4:** Design-Phase abgeschlossen (Use-Cases, Klassenmodell, Architektur)
3. **Woche 8:** Alle Kernfunktionen implementiert
4. **Woche 10:** System getestet und stabil
5. **Woche 12:** Projekt vollständig dokumentiert und abgegeben

6.2 Erfolgskriterien

Das Projekt gilt als erfolgreich, wenn:

- Das Spiel mindestens 2 funktionierende Level hat
- Phishing-E-Mails korrekt erkannt werden
- Ein Highscore-System mit Sterne-Bewertung implementiert ist
- Die Anwendung als ausführbare JAR-Datei läuft
- Eine vollständige Dokumentation (Javadoc, README, UML) vorliegt
- Das Projekt termingerecht abgegeben wird

7 Überwachung und Berichtsmethoden

7.1 Fortschrittskontrolle

- **Wöchentliche Selbstreflexion:** Überprüfung des Fortschritts anhand der Meilensteine
- **Git-Commits:** Regelmäßige Commits mit aussagekräftigen Commit-Messages
- **Task-Board:** Verwendung von GitHub Issues oder Trello zur Aufgabenverfolgung

7.2 Anpassungen

Der Zeitplan kann bei Bedarf angepasst werden, wenn:

- Unvorhergesehene technische Probleme auftreten
- Sich Anforderungen ändern
- Verzögerungen durch externe Faktoren entstehen

Bei größeren Abweichungen werden Features priorisiert (MVP-Ansatz: Minimum Viable Product zuerst fertigstellen).

8 Fazit

Dieser Zeitplan dient als **Leitfaden für die strukturierte Entwicklung** des Phishing Defender Spiels über einen Zeitraum von 12 Wochen. Durch die klare Definition von Aufgaben, Meilensteinen und Risiken wird sichergestellt, dass das Projekt termingerecht und mit hoher Qualität abgeschlossen werden kann.

Die **flexible Gestaltung** des Plans ermöglicht es, auf unvorhergesehene Herausforderungen zu reagieren, während die definierten Erfolgskriterien einen klaren Rahmen für die Bewertung des Projekterfolgs bieten.

Projektziel:

Ein funktionsfähiges, gut dokumentiertes Lernspiel zur Phishing-Erkennung, das als ausführbare JAR-Datei läuft und termingerecht abgegeben wird.

Der vorliegende Zeitplan bildet die Grundlage für ein erfolgreiches Projektmanagement und stellt sicher, dass alle wichtigen Aspekte der Softwareentwicklung – von der Anforderungsanalyse über die Implementierung bis hin zur Dokumentation – systematisch und nachvollziehbar durchgeführt werden.

Ende der Dokumentation