Verslag Python

Salah Jaafar

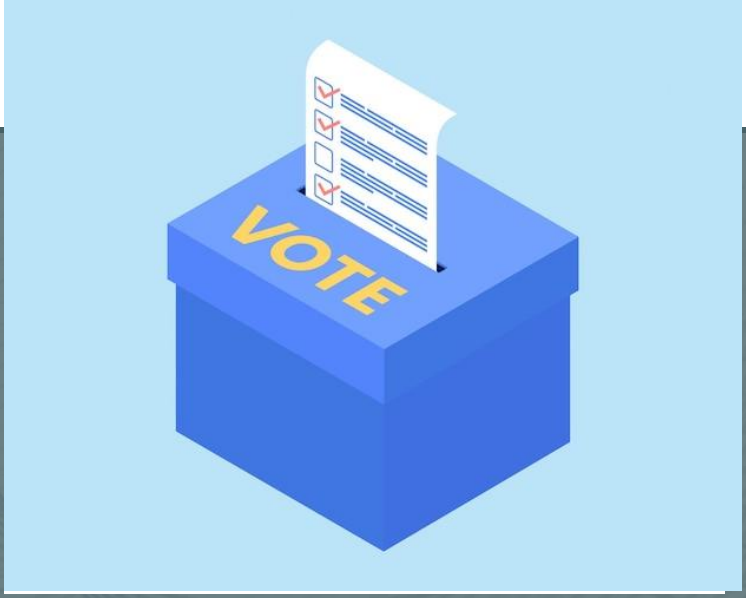
1ITCSC3

—

2023-24

Voorwoord

Dit rapport beschrijft de ontwikkeling van een kiessysteem dat kiezers de mogelijkheid biedt om te stemmen op verschillende partijen/lijsten en tegelijkertijd de zetelverdeling berekent en presenteert. De verschillende onderdelen van het systeem omvatten kiezers, lijsten, stemcomputers, een stembus en een scanner. Bovendien zijn er nuttige functies en methoden gemaakt voor de verdeling van zitplaatsen en het genereren van HTML-output. Dit rapport beschrijft de stappen en beslissingen die zijn genomen tijdens het ontwikkelproces.



Fase 1: Analyse en Ontwerp

Het analyseren van de eisen en het ontwerpen van de systeemstructuur waren de eerste stappen van het proces. De primaire eisen waren:

- Het uitvoeren van een simulatie van kiezers die stemmen uitbrengen.
- Het bijhouden van de stemmen per lijst.
- Om de stembiljetten te controleren.
- Calculeren van de verdeling van de zetels op basis van het aantal stemmen.
- Het maken van een HTML-rapport dat de verdeling van zetels weergeeft.
- Op basis van deze specificaties werden de volgende klassen vastgesteld

Op basis van deze specificaties werden de volgende klassen vastgesteld:

1. De kiezer: mensen die stemmen.
2. De kandidaat: mensen die stemmen krijgen.
3. De lijst: Stellen de partijen voor
4. Stembiljet: De stem van de kiezer
5. Stembus: Plaat waar alle stemmen bewaard worden.
6. Scanner: Controleren van stembiljetten.
7. StemComputer: Tool om te stemmen
8. Het USB-stick: Simulaties uitvoeren met behulp van een USB-stick.
9. Chipkaart: Simulatie van kiezerschipkaarten
10. Het kiessysteem: Beheer van het kiesproces.
11. Utilities: Berekenen van zetelverdeling en genereren van HTML-output.

Fase 2: ontwikkelingsfase

Hieronder heb ik mijn ontwikkelingsproces beschrijven:

Ik begon met het maken van de nodige klasse. Elke klasse had eigen attributen nodig bijvoorbeeld: naam, leeftijd en opstartcode etc. Sommige van deze klassen hadden wat functionaliteit nodig bijvoorbeeld de klasse Lijst had een methode nodig om kandidaten te toevoegen, dus ik zorgde ervoor dat elke klasse zijn eigen methoden heeft. Het moeilijkste gedeelte was de klasse Kiessysteem. Ik moest heel nauwkeurig zijn met iedere methoden en toepassing, wat me heel veel tijd koste. Het combineren van meerdere klassen in een klasse zorg ervoor dat ik heel veel errors krijg. Het enige oplossing was dat ik elke methode apart moest testen en de resultaten checken en zijn of dat wat ik zou verwachten.

Fase 3: Implementatie

Elke klasse werd in een ander Python-bestand geplaatst. Dit maakte de codebase eenvoudig te begrijpen en onderhoudbaar. Elke module werd afzonderlijk ontwikkeld en getest terwijl de implementatie iteratief werd uitgevoerd.

De kiezer en de lijst:

Namen, leeftijd en stemstatus van kiezers vormen de kiezer-klasse.
Klasse lijst voor Partijen met kandidaten en aantal stemmen.

Kandidaat:

Klasse om een kandidaat te voorstellen.

Stemcomputers, stembiljets en stembus:

StemComputer-klasse om het stemproces uit te voeren.
Klasse van stembiljetten
Klasse voor het verzamelen van stembiljetten.

Scanner en Chipkaart**:**

Scannerklasse om stembiljetten te scannen.
Chipkaartklasse voor kiezers

Het kiessysteem:

Klasse voor de simulatie → Hierin worden alle andere klassen geïmplementeerd + Html-output wordt hier gemaakt, de verdeling van zetels

Fase 4: Testen en verifiëren

Ik heb de modules allemaal apart getest om ervoor te zorgen dat het correct werkte. En dat kon ik doen via het importeren van unittest module. Ik heb ook wat andere technieken gebruik die ik op het internet vond. Alle modules heb ik in een aparte file gezet om mijn code leesbaar te houden.

Fase 5: Problemen

1. Tijdens het maken en testen van het systeem in zijn geheel had ik problemen met het genereren van fake data voor mijn kiezer. Daardoor heb ik besloten om het toch via een random.choise() op een array te doen. Ik heb op het internet gezocht of dat het anders kan, maar het was te ingewikkeld.

2. Andere probleem dat ik ondervond, is dat het stem systeem was niet random genoeg. Het deed wat er gevraag wordt maar de output was bijna altijd gelijk, dus ik heb gebruik gemaakt van gewichten op de lijsten/partijen, dat zorgt ervoor dat er altijd een partij meer gekozen werd dan anderen, dat maakte het systeem meer onvoorspelbaar.

3. Terminal output was onduidelijk en veel tekst onder elkaar, dus ik heb een soort van kleur toe gevoegd om het proper te houden.

Fase 6: Refactoring

Na dat ik alles heb getest, probeerde ik mijn code netjes te schrijven en te verbeteren aan de hand van list comprehensie waar nodig was. Bijvoorbeeld voor het initialiseren van chipkaarten naar een conventionele For-loop

Resultaten

Ik heb een volledig functionerend kiessysteem gemaakt na een gestructureerd ontwikkelingsproces en uitgebreide testfasen. De belangrijkste resultaten zijn:

- Module design: Elke klasse is in een ander Python-bestand geplaatst, waardoor de code gemakkelijk te begrijpen en onderhoudbaar is.
- Klassefuncties: De belangrijkste categorieën (Kiezer, Lijst, Kandidaat, Stemcomputers, Stembiljets, Stembus, Scanner, Chipkaart en Kiessysteem) zijn allemaal succesvol toegepast met de benodigde functies en technieken.
- Tests: De betrouwbaarheid van de code werd verhoogd doordat elke module afzonderlijk werd getest met behulp van de unittestmodule.
- Oplossing voor het genereren van gegevens: Het gebruik van random.choice heeft de problemen opgelost met het maken van valse kiezersgegevens.
- Verbeterde Randomisatie: Het stemsysteem is aangepast met gewichten om de uitkomsten onvoorspelbaarder te maken, wat resulteerde in een realistischere simulatie.

Het einde product is een werkende stemsysteem die onderhoudbaar, effectief en maakt realistische simulaties.

Veiligheidsreflectie

De ontwikkeling van een kiessysteem vereist veiligheid om het stemproces veilig te houden, de anonimiteit van kiezers te beschermen en beveiliging te bieden tegen aanvallen en manipulatie. Dit gedeelte onderzoekt de veiligheid van het gekozen kiessysteem, vindt problemen en geeft suggesties voor verbeteringen.

Kwetsbaarheden en Verbeteringen

Manipulatie van stemmen:

Potentiële risico's: Cryptografische waarborgen ontbreken; Stemmen worden rechtstreeks toegewezen aan lijsten.

Verbetering: Implementeren van end-to-end encryptie voor stemmen die zijn versleuteld

Onvoldoende authenticatie en autorisatie:

Potentiële risico's: eenvoudige items zoals USB-sticks en chipkaarten die geen sterke authenticatie hebben

Verbetering: gebruik van digitale certificaten, 2-factorauthenticatie of biometrische verificatie

Integriteit van stemcomputers:

Potentiële risico's: potentieel kwetsbaar voor malware of ongeautoriseerde toegang

Verbetering: veilig booten en regelmatige integriteitscontroles.

Privacy van kiezers:

Potentiële risico's: Mogelijkheid om stemgedrag te volgen via metadata.

Verbetering: strikt beleid voor het behouden van gegevens en het gebruik van homomorfe encryptie

Fysieke veiligheid:

Potentiële risico's: Fysieke delen kunnen worden gestolen of gesaboteerd.

Verbetering: Verzegelde stemcomputers, beveiligde opslag en camera's in stemlokalen

Conclusie

Hoewel het stemsysteem een verkiezingssimulaties biedt, bevat het wel wat securityproblemen. De veiligheid van zo een systeem is belangrijk om de eerlijkheid en vertrouw te garanderen. Het implementeren van sterke encryptie technieken, fysieke beveiligingsmaatregelen en sterkere authenticatie- en autorisatiemethoden is dus essentieel.