1 Casus

Je voert als team een Scrum-project uit waarbij een complexe webapplicatie met database wordt opgeleverd. In dit project werk je verder met dezelfde Casus als bij het Security Project (SPR), maar de juiste requirements worden aangeleverd. De webapplicatie wordt gemaakt met ASP.NET Core 3.1.

1.1 Beschrijving webapplicatie

De Gemeente Den Haag wil voor haar burgers een webapplicatie laten ontwikkelen waarmee burgers
(1) problemen kunnen melden in de wijk met een omschrijving, eventueel een foto en een voorgestelde oplossing.
(2) Vervolgens kunnen buurtbewoner stemmen voor de verschillende oplossingen
(3) kunnen chatten en
(4) een misdaad kunnen melden.

1.1.1 "Publieke" problemen

In een "publiek" gedeelte van de app kunnen burgers samen werken aan een leefbare wijk. Hierin kunnen ze publieke meldingen over zwerfvuil, enge plekken, losliggende stoeptegels, tekort aan parkeerplekken, bomen die voor overlast zorgen, egels in nood etc.) melden en bespreken in een chat.

Voorwaarden voor een **publiek probleem** om gemeld te kunnen worden:

- De gemeente moet er iets aan kunnen doen, dus "100 rijden op de snelweg is te langzaam" is geen relevante melding.
- Het moet een systematisch probleem zijn, dus "de muziek van de buren staat te hard" mag niet.
- Het moet een probleem zijn dat meerder burgers aangaat.
- Alleen burgers in de eigen wijk kunnen meediscussiëren.

Voldoet een probleem niet aan deze voorwaarden, dan zal het moeten worden verwijderd door een moderator. De indiener moet dit terug kunnen zien bij zijn meldingen inclusief de onderbouwing voor de verwijdering.

1.1.2 "Gevoelige" meldingen of klachten

In een "anoniem" gedeelte moeten de meer gevoelige onderwerpen kunnen worden besproken zoals wiet plantages, buurman slaat de buurvrouw of andersom, hangjongeren met naam en toenaam etc. Het is hier uiterst van belang dat zowel de identiteit van de melder als het onderwerp van de melding goed afgeschermd blijven.

De gebruiker kan de meldingen anoniem en zonder inhoud terugvinden in een overzicht om de gebruiker de bevestiging te geven dat de melding aangekomen is.

1.1.3 Aanmelden

Om een melding te kunnen maken via de webapplicatie maakt de burger eenmalig een account aan met naam en woonadres. Zonder account is het maken van een melding niet mogelijk.

De chat 'staat open' voor alle burgers met een account om meldingen te bespreken.
Bij misbruik (bedreigingen, schelden e.d.) moet het mogelijk zijn de chat door een moderator te laten verwijderen en verdere toegang van de gebruiker tot de chat te blokkeren. Een gebruiker moet daarom ook misbruik kunnen melden bij de moderator.

Voor het melden van een klacht in het **anonieme gedeelte** moet rekening gehouden worden met de privacy van de melders.
Na de aanmelding voor het publieke gedeelte, krijgt iedereen een nummer. Met dat nummer kan men de anonieme aanmelding doen. Dat nummer wordt in de databank m.b.v. PETS ontkoppeld met de melding. Slechts geautoriseerde medewerkers met rechten kunnen de koppeling maken.
Het is van groot belang dat burgers erop kunnen vertrouwen dat hun persoonlijke gegevens veilig zijn en ook dat dit ook duidelijk aangegeven is op de site.

2 Requirements

2.1 Functionele systeemeisen

2.1.1 Inloggen

- Op elke pagina moet te zien zijn of en **welke gebruiker is ingelogd**.
- Op elke pagina bevindt zich een **inlog-knop** en een **registreer-knop** als er nog niet is ingelogd.
- Na het x keer onjuist inloggen (per IP), moet er een **captcha** worden opgelost bij elke volgende log-in.
- Na het x keer onjuist inloggen (per gebruiker), wordt er **gesuggereerd op "Wachtwoord vergeten"** te klikken.
- Als er geen gebruikersnaam/wachtwoord is ingevuld dan wordt er een **foutmelding** gegeven
- Als er op wachtwoord vergeten wordt geklikt, dan wordt er een **wachtwoord herstel mailtje** gestuurd met een link naar een pagina waar het wachtwoord opnieuw kan worden ingesteld. Deze link **expiret** na een aantal dagen.
- Stuur een emailtje naar een gebruiker als deze inlogt van een **ongebruikelijke device of locatie**.
- Het systeem moet over **elke inlogpoging even lang doen** om er zo voor te zorgen dat het niet duidelijk is hoe ver in het authenticatieproces is gekomen.

2.1.2 Registreren

- **Valideer client-side** of alle velden (email, huisnummer, toevoeging, etc.) goed zijn ingevuld.
- Implementeer **adresvalidatie**.
- Gebruik bij registreren een **captcha**.
- Wordt na registeren doorverwezen naar een **email-verificatie-pagina** of pagina waar wordt uitgelegd dat er per post een verificatie gedaan wordt.
- Na het inloggen voor de eerste keer wordt een **rondleiding** door de website gegeven, waarin buttons worden gehighlight met uitleg daarover. Aan het einde van deze tour (die ook geskipt kan worden), wordt er naar de profielpagina verwezen, waarop de gebruiker

zijn gegevens kan controleren en aanpassen, en mogelijk ook een profielfoto kan instellen. Dit kan ook met een uitlegpagina.
- Op de profielpagina is ook een knop waarmee alle **data opgehaald** kan worden die is opgeslagen door het systeem. Alsmede een knop waarmee een gebruiker het hele account kan **opheffen**.
- **Moderators** worden door de admin aangewezen, en er hoeft dus geen front-end gemaakt te worden om moderators aan te wijzen.

2.1.3 Melding bekijken

- De **titel** en **beschrijving** moeten zichtbaar zijn. Ook de **auteur** de melding heeft gedaan is zichtbaar.
- Het aantal **likes** moet zichtbaar zijn.
- De gebruiker kan (maximaal 1 keer) op **+1** klikken.
- De gebruiker kan op **report** klikken als de melding bijvoorbeeld aanstootgevend is. Er wordt dan een melding gedaan naar de moderator.
- Meldingen worden **automatisch gesloten** na 30 dagen geen reactie.
- Er kunnen **reacties** geplaatst worden on een melding.
- Gevoelige gegevens die in de reacties worden gedeeld kunnen door een moderator **gecensord** worden.
- Per reactie is zichtbaar waneer deze is geplaatst en door wie.
- Subcomments bestaan niet.
- Een moderator kan een melding **sluiten** als de oplossing gevonden is / er niet meer over wordt gepraat / om een andere reden geen extra reacties meer mag hebben.

2.1.4 Overzicht publieke meldingen

- Er moet **gesorteerd** kunnen worden op aantal views, aantal likes, en datum.
- De gebruiker kan zoektermen invullen om specifieke meldingen te **zoeken**.
- De gebruiker kan een **datum range** aangeven waarbinnen de meldingen gepost moeten zijn.
- De gebruiker kan er ook voor kiezen te **filteren** op meldingen waarop de gebruiker heeft geliket.
- De resultaten moeten **per 10** op een pagina te zien zijn, met een knop kan dan de volgende 10 worden bekeken.
- Per melding is de titel en een **gedeelte van de inhoud** te lezen als voorbeeld. Ook is het aantal likes en views te zien.
- Standaard worden alle open meldingen weergegeven, maar de gebruiker kan er ook voor kiezen gesloten meldingen ook zichtbaar te maken.

2.1.5 Melding maken

- **De melding kan anoniem gedaan worden of publiekelijk**.
- Er moet een **categorie** kunnen worden ingesteld
- De **tekst en titel** moeten worden aangepast
- Als de titel lijkt op de titel van een bestaande melding, dan moet een waarschuwing gegeven worden
- Als de melding minder dan 10 karakters bevat of de titel minder dan 3, dan moet een foutmelding gegeven worden.

- Na het aanmaken van de melding wordt de gebruiker naar het overzicht geleid.
- Het moet mogelijk zijn een **foto up te loaden** die bij de melding hoort.
- Bonus: Met AI algoritmen kunnen blote en bloederige fotos's gefiltert worden.

2.2 Niet-functionele systeemeisen

2.2.1 Veiligheid

- De data wordt in een ge-encrypte database opgeslagen.
- Niet alle medewerkers hebben toegang tot alle onderdelen van de database.
- De gebruikers worden geanonimiseerd opgeslagen. Bij de tabel met persoonsgegevens kunnen alleen medewerkers met veel rechten, bijvoorbeeld de database admin.
- Bij een fout wordt er aan de eindgebruiker zo min mogelijk details getoond over de fout.
- De website is cross-site-scripting-attack-veilig.
- De persoonsgegevens van een gebruiker zijn alleen zichtbaar/aanpasbaar voor de diegene die wegens zijn rol de juiste rechten heeft gekregen.
- Beveiligingsniveau: de verwerker dient een passend beveiligingsniveau te waarborgen (o.a. pseudonimiseren en versleutelen van persoonsgegevens)
- De wachtwoord hashes worden salted opgeslagen in de database.
- Het systeem is bestand tegen Broken Session Management
- Bekende kwaadwillenden hun IP adressen worden opgeslagen in een Black list.

2.2.2 Privacy

- Alleen relevante persoonsgegevens (naam, adress, email-adres) worden opgeslagen (gegevensbeperking)
- De gebruiker moet ten alle tijden bij de persoonsgegevens kunnen in een profielpagina
- Het systeem moet in staat kunnen zijn gebruikersdata te verwijderen wanneer men het account heeft verwijderd
- Het systeem verwijdert (deel)gegevens (bijvoorbeeld logs) die niet langer nodig zijn na drie maanden.
- De persoonlijke gegevens moeten gescheiden worden opgeslagen van de anonieme meldingen.
- Het systeem moet alle persoonsgegevens in de database versleutelen. Dit moet gebeuren op de ASP.NET applicatie server dus voordat het verstuurd wordt naar de database (verberg).
- Het systeem moet in staat zijn om persoonsgegevens aan te passen of te verwijderen (controle)
- Het systeem moet in staat zijn om aan de gebruikers te tonen waarvoor de gegevens worden verwerkt en voor welke doelen (inform).
- Het systeem moet in staat zijn om de bepaalde gebeurtenissen te loggen met verschillende niveau van urgentie (toon aan).
- Privacy by design en default dienen standaard te worden geïmplementeerd

2.2.3 Usability

- De website is handig in gebruik en de knoppen staan overzichtelijk.
- Alle afbeeldingen hebben een alternatieve omschrijving.
- Het systeem moet te gebruiken zijn voor alle mensen met een beperking.
- Het systeem is functioneel op elk platform en device

- Het systeem verliest geen essentiele informatie bij het aanpassen naar kleinere devices
- De website moet een fall back font hebben.
- Er moet een makkelijk taalniveau worden toegepast
- De tekst op we website heeft een minimum contrast ratio van 4,5:1

2.2.4 Performance

- De website reageert niet zo langzaam dat gebruikers geirriteerd worden.
- Er is iets bekend over de schaalbaarheid van de applicatie; hoe verhoudt de laadtijd van een pagina zich met het aantal meldingen of het aantal reacties?

2.2.5 Overig

- Het systeem is voldoende gedocumenteerd en de code bevat genoeg commentaar (en niet te veel), zodat andere ontwikkelaars het kunnen doorontwikkelen.
- Het systeem is ontworpen met bekende architectuur- en ontwerp patronen, zodat het uitbreidbaar en onderhoudbaar is.
- Het systeem moet de gebruikte gegevens van de gebruiker kunnen sturen naar de gebruiker.

3 Eindproduct

Aan het einde van het project levert het team naast de gemaakte website (de code op Github en gehost in Azure), ook een een architectuurdocument op. Dit document beschrijft de code voor mede-ontwikkelaars (en mogelijk toekomstige ontwikkelaars). Dat het agile manifesto “working software over comprehensive documentation” prefereert, betekent niet dat documentatie altijd slecht is.

- Voeg modellen (UML diagrammen) toe uit de analyse fase van het ontwikkeltraject, die het probleemdomein beschrijven, en uit de ontwerp fase, die het softwaresysteem modelleren.
- Noem de verantwoordelijkheden van klassen en de afhankelijkheden ertussen.
- Voeg de site-map en wireframes van de front-end toe die zijn gemaakt zijn voor de implementatie in HTML/CSS/JS.
- Noem en verantwoord de keuzes die gemaakt zijn in de architectuur van de webapplicatie.
- Voeg een testplan toe die beschrijft *welke* onderdelen getest gaan worden en *waarom* bepaalde aspecten in het bijzonder uitvoerig getest moeten worden. Dit document beschrijft unit-testen, integratie testen, maar ook bijvoorbeeld testen voor niet-functionele systeemeisen (kwaliteit van de code testen).