Løsningen har fulgt kravene til oppgaveteksten og består av tre sider; Forside, spillside og highscoreside. Løsningen har vært utviklet på en Google Nexus 5 og burde ikke behøve noe konfiguering før oppstart. Løsningen har mininum SDK-versjon 23(Marshmallow), jeg ser i retrospekt at jeg burde ha gått et par versjoner lengre bak for å få en større målgruppe. Det er ikke ført kilder/referanser på dette prosjektet siden det ikke har vært **en** spesifikk kilde som har påvirket prosjektet i noe særlig grad, men heller mange små for å hjelpe meg med småproblemer.

Under ser du begrunnelse for valgene jeg har gjort.

Generelt om produktet: Generelt føler jeg koden gjør det den skal, er lesbar og at layouten er intuitiv. Smart bruk av forløkker gjør at det blir lettere å endre prosjektet i etterkant, med unntak av checkBoard-metoden. Layouten kunne vært finere, og jeg kunne ha gått mer utenfor oppgaven for å heve den, men jeg har fått inntrykk av at fokuset er solid og lesbar kode og satte heller fokuset mitt der. Det er litt småpirk her og der, men jeg har i det minste begrunnet hvorfor det er slik, og føler ikke det påvirker applikasjonen min noe særlig negativt.

FrontPage: Jeg har satt en begrensning på antall bokstaver I tekstboksene slik at brukerne ikke kan bli altfor liberale med brukernavnene sine. Blir det lengre enn 11 bokstaver kan det potensielt påvirke layouten i de andre sidene, og jeg ser ikke behovet for å ha et mye lengre brukernavn.

GameBoard: Jeg har satt vekt på at det ikke skal være logikk i activityen utenom navigering og instansiering. Klassen er tett sveiset med BoardLogic, og sender med sin egen activity ved instansieringen av BoardLogic for at det ikke skal bli veldig mye frem og tilbake mellom klassene når UI skal endres. En svakhet er at spillerene ikke kan velge hvem som vil starte, men jeg syntes ikke dette var funksjonalitet som var viktig for applikasjonen.

BoardLogic: checkBoard-metoden kunne ha blitt implementert bedre, hvor den nå ikke er skalérbar og har høy duplikasjon. Jeg vurderte å bruke magic square algoritmen, men jeg følte at det ville blitt mye kode uten å forbedre noe særlig funksjonalitet. Metoden min trenger bare å sjekke status på brettet fem ganger, og er ikke veldig kostbar. Jeg har også sørget for at den skulle bli mest mulig lesbar ved å kommentere hva hver sjekk sjekker.

BoardButtonListener: Jeg lagde denne lytterklassen fordi da kan jeg lett sette lyttere på knappene i en for-løkke med riktig parametre.

Player: Ment som en enkel POJO, men jeg la til en hjelpemetode med hensikt til å øke lesbarhet i BoardLogic.gameOver.

Highscore: Jeg valgte å bruke ListView fordi den er dynamisk og det er highscore-listen også. Jeg måtte derimot ofre estetikk på grunn av dette, men da dette ikke var fokuset i oppgaven lot jeg det være slik.

DBHandler: Det kan bli en del kall mot databasen på grunn av hjelpemetodene som er der, dog det hjelper på lesbarheten kan det koste litt. Jeg kunne ha lagret mer data i lokale variabler, men da hadde det blitt dyrere å starte appen. Siden levetiden til appen er (antageligvis) kort valgte jeg å heller ha fokus på mer lesbar kode. Databasen vil aldri ha flere enn 10 rader så da blir heller ikke kallene like kostbare som det ville vært på andre databaser.