Seoses veebi arendustehnoloogiate olulise täienemisega viimastel aastatel, on mitmekordselt suurenenud ka veebi arendamise kiirus. Kuna arendamise käigus insenerid ja arendajad paratamatult teevad ka vigu, siis tänu arenduse kiirusele ja kogusele on kordades suurenenud ka tekkivate vigade arv. Neid veasituatsioone esile tuua ning vigu käsitsi kontrollida aga võtab väga palju aega – ressurss, mida ettevõtetel tihtipeale üle pole. Aja kokkuhoiu ning vigade eemaldamise eesmärgil on loodud mitmeid erinevaid liideseid ja programme selleks, et testimist hõlbustada, eelkõige automatiseerimise näol.

Autor valis lõputöö teemaks oma tööandja, AS SEB Panga internetipanga maksevormi testide vajaduste ja testilugude kaardistamise ning testide kirjutamise. Mõte teema valikuks tekkis huvist testimise ning testide kirjutamise kohta eelkõige selle tõttu, et autori õppekavas seda teemat puudutatakse vähe, ning panga vajadusest maksevormi testimine automatiseerida.

Kuna maksevorm ja sellega seotud toimingud on üks kõige enam kasutatud funktsionaalsuseid internetipangas, siis on äriliselt väga oluline, et see töötaks alati vigadeta ning ei põhjustaks kasutajale ebameeldivusi. Hetkel käsitsi läbi tehtavad testid võivad lühiperioodil võtta küll mitte märkimisväärselt palju aega, kuid kui pärast iga uuendust on vaja käsitsi teste läbi teha, võtab see pikal perioodil väga palju väärtuslikku aega ära, mida saaks hoopis maksevormi paremaks arendamisel ära kasutada.

Autor kasutas automaattestide kirjutamisel varasemaid teadmisi AS SEB Panga maksevormi iseärasustest ning testide automatiseerimisest. Seleniumi raamistikus kasutati automaattestide kirjutamiseks Seleniumi WebDriver liidest koos JUnit testimise raamistikuga. Et võrrelda klassikaliselt tihti valitud Seleniumi raamistikku mõne teise meetodiga, proovis autor pooli automaatteste kirjutada CasperJS raamistikku kasutades, andes võimaluse mõlemaid piisavas ulatuses võrrelda.

Lõputöö esimene pool sisaldab ülevaadet testimisest ja selle automatiseerimisest. Autor selgitab käsitsi ning automatiseeritud testimise põhipunkte ning võrdleb neid kahte omavahel – toob välja mõlema eelised ja puudused ning kirjeldab võimalikke lahendusi puuduste vältimiseks. Töö teises pooles kirjeldati testilugusid, nende koostamise protsessi ning toodi ka välja koostatud testilood. Lisaks kirjeldab autor ka AS SEB Panga internetipanga maksevormi ning selle iseärasusi ja tingimusi testide täitmiseks. Autor toob välja vajalikud tehnoloogiad ning kirjeldab põgusalt lõputöö käigus vaja läinud kasutusi igaühele. Töö kokkuvõttes toodi välja

saavutatud tulemused, võrreldi kahte kasutatud testimise meetodit ning anti ülevaate mõlemast antud ülesannete täitmiseks.

Lõputöö üks peamiseid eesmärke oli AS SEB Panga erakliendi internetipanga maksevormi seniste käsitsi testide asemele arendada automaattestid, mis vähendaksid korduvale testimisele kuluvat aega ning lubaks arendajatel rohkem fokusseerida arendusele, mitte arendusest tingitud vigade tagaotsimisele. Lõputöö tulemusena sai loodud 10 automaattesti Seleniumi ja JUnit raamistikke kasutades, mis katavad internetipanga maksevormi olulisemad funktsionaalsused. Testimistulemusi on võimalik näidata korraga kogu arendusmeeskonnale, mitte ainult ühele testijale, lubades kogu meeskonnal parandusprotsessis paremini kaasa lüüa. Automaattestide käivitamise tulemusena luuakse raportid, mis annavad asjalikku tagasisidet ning aitab seeläbi vigu kiiremini parandada.

Lisaks sai CasperJS raamistikku kasutades loodud 5 automaattesti, millega on võimalik võrrelda kahte testimismeetodit piisavas ulatuses. Nende viie testi tulemusena on autor järeldanud, et kuigi JavaScript keeles testimine on üsna uudne ning tänasel päeval mitte väga laia kasutust leidev meetod, on see paljulubav ning annab testijale mitmeid uusi tööriistu, millega testimisprotsessi kergendada ja tulemuslikkust tõsta.

Võrdluse tulemusena leiab autor, et kui AS SEB Pank toetaks oma testkeskkonnas JavaScript testimise tehnilise võimalusi, oleks võimalik lisaks Seleniumi raamistikuga võidetud ajale ja kindlusele veel rohkem aega testimisele kuluvalt ajalt kärpida. JavaScriptis teste kirjutada võib esialgu tunduda veider, kuid tegelikult on see mugav ja väga võimalusterikas arendajale, kes seda keelt juba varasemast tunneb.