Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Машинско инжењерство

Ниво студија: Основне академске студије

Модул: Информатика у инжењерству

Предмет: Софтверски инжењеринг

Број индекса: 40/2014

Марјан М. Спасојевић

Апликација за исцртавање графика функције

завршни рад

Комисија за преглед и одбрану:

1. **др Ненад Филиповић - ментор**
2. _____
3. _____

Датум одбране:_____

Оцена:_____

Садржај

У оквиру овог завршног рада кандидат треба да уради апликацију која по уносу функције од стране корисника исцртава одговарајући график функције. Апликација је урађена у *JavaScript*-u.

Препоручена литература:

[1] Jay Sridhar; "What Is JavaScript and How Does It Work? " интернет:https://www.makeuseof.com/tag/what-is-javascript/

[2] "Guide for using SASS preprocesing"; интернет: https://sass-lang.com/guide

[3] Филиповић Н.: *Алгоритми и структуре података,* Универзитет у Крагујевцу, Факултет инжењерских наука, Крагујевац, 2010.

Крагујевац, октобар 2018.

ментор:
др Ненад Филиповић

_____

Резиме

У овом раду описана је апликација за исцртавање графика математичких функција. У првом делу рада дат је увид у саме технологије коришћене за развој апликације. Затим, описан је сам развојни процес апликације, дат је увид у саму структуру података и описан је принцип рада важнијих функција. На крају дат је опис самог интерфејса апликације као и упуство за коришћење.

Кључне речи: график функције, решење функције, математичке функције.

Abstract

This paper describes an application for graphical presenting mathematical functions. In the first part of the work, the insight into the application development technology is given. Then, the application development process is described , also data structure itself is given and the principle of important functions is described. At the end , description of the application interface as well as user manual is given.

Keywords: draw function graph, function solution, mathematical function.

1. Увод

Кроз овај рад биће описан развој апликације за исцртавање графика математичких функција. Као развојно окружење, при развоју ове апликације, коришћен је програм **VISUAL STUDIO CODE,** ово је веома популаран програм Microsoft корпорације, и налази велику примену у развоју различитог web садржаја. Као развојни програм при изради ове апликације изабран је **JAVASCRIPT (**у даљем тексту JS**)** језик. Поред JS-а при изради апликације користе се основни језици за сам приказ било каквог садржаја на web-у, то су наравно **HTML** і **CSS**.

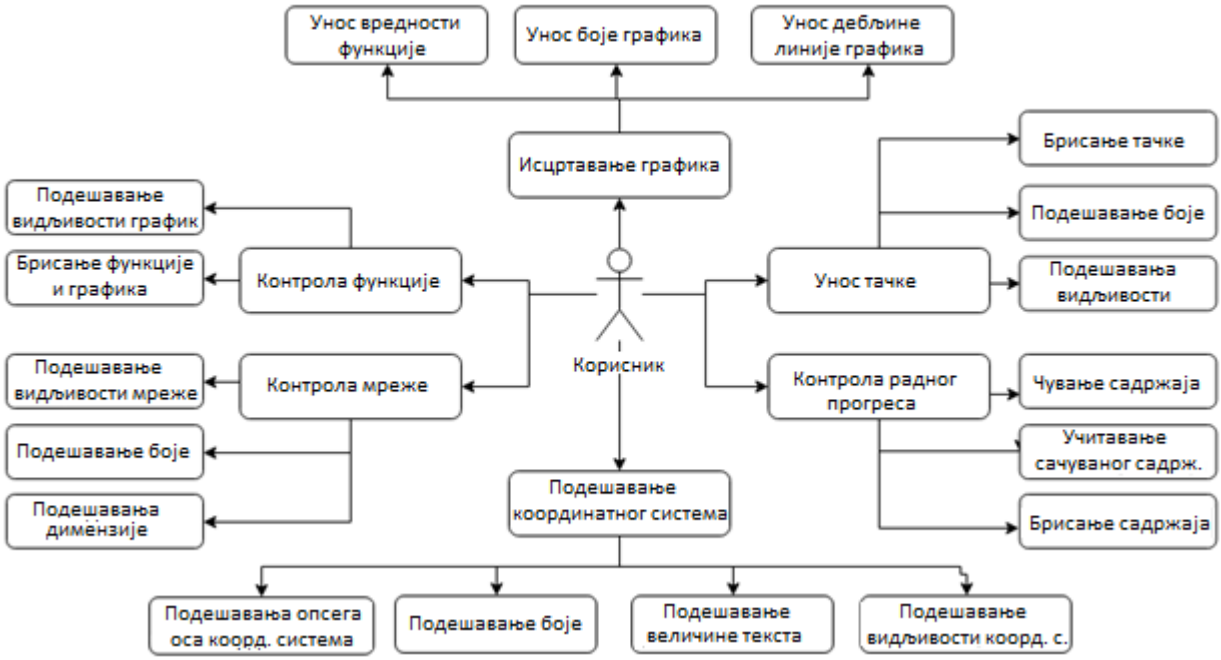
Чињеница је да су многи, који не поседују довољно информација, склони да наведу да су JAVA и JavaScript потпуно истоветни програмски језици, што је апсолутно погрешно, овом приликом ће се прецизирати шта се тачно подразумева под одредницом JS. У питању је такозвани објектни скриптни језик, чија је основна сврха коришћења везана за побољшање динамике одређене странице. Заправо је ствар у томе да када се користи HTML код, његова улога је само да обликује одређене елементе странице, као што су рецимо текст, табеле, линкови или форма, и да их након тога уреди, али овај код нема улогу да одреди на који начин ће се конкретни елементи понашати. Управо тада се у читав процесу укључује JS, који има значајну улогу у креирању понашања ових елемената. А када се он комбинује са HTML - ом и CSS - ом, тада настаје такозвани динамички HTML (**DHTML**). Када се каже да је неки скриптни језик објектни, то заправо значи да особа која ради са њим треба да одреди и тип података, али и да дефинише која ће тачно функција бити примењена, а што се мора прецизирати на основу структуре конкретних података. Такав принцип доводи до тога да структура одређених података уствари почиње да буде третирана као објекат у оквиру кога се налазе и ти подаци и одређене функције. Коришћењем овог програмског језика се такође одређују и односи између два објекта. Иако JS не спада у ООЈ (Објектно Орјентисане Језике) у скорије време у JS се уводи структура података специфична за ООЈ тзв. **КЛАСЕ** што омогућава JS коду да буде уређен много више налик **JAVI** и **C++** језику тј. у објектно орјентисаном смислу.

Како би се сам развојни поступак апликације убрзао за уређивање (стилизовање) **HTML** садржаја коришћен је **SASS**. SASS је CSS препроцесор – слој између стилова којима се управља и .цсс фајлова који се прослеђују серверу. SASS попуњава рупе у CSS - у као језику, дозвољавајући писање DRY код који ће бити бржи, ефикаснији и лакши за одржавање. SASS омогућава додавање супер функционалности класичном CSS -у, а затим га преводи (компајлира) у регуларне CSS фајлове преко различитих програма или web framework plugin-а. Крајњи циљ развоја ове апликације јесте да кориснику пружи алат за истовремено исцртавање више графика различитих математичких функција и да се са сваком функцијом може понаособ манипулисати. Такође циљ апликације јесте да кориснику пружи задовољавајући UX (user experience) и пре свега апликацију једноставну за коришћење.

2. UML дијаграми

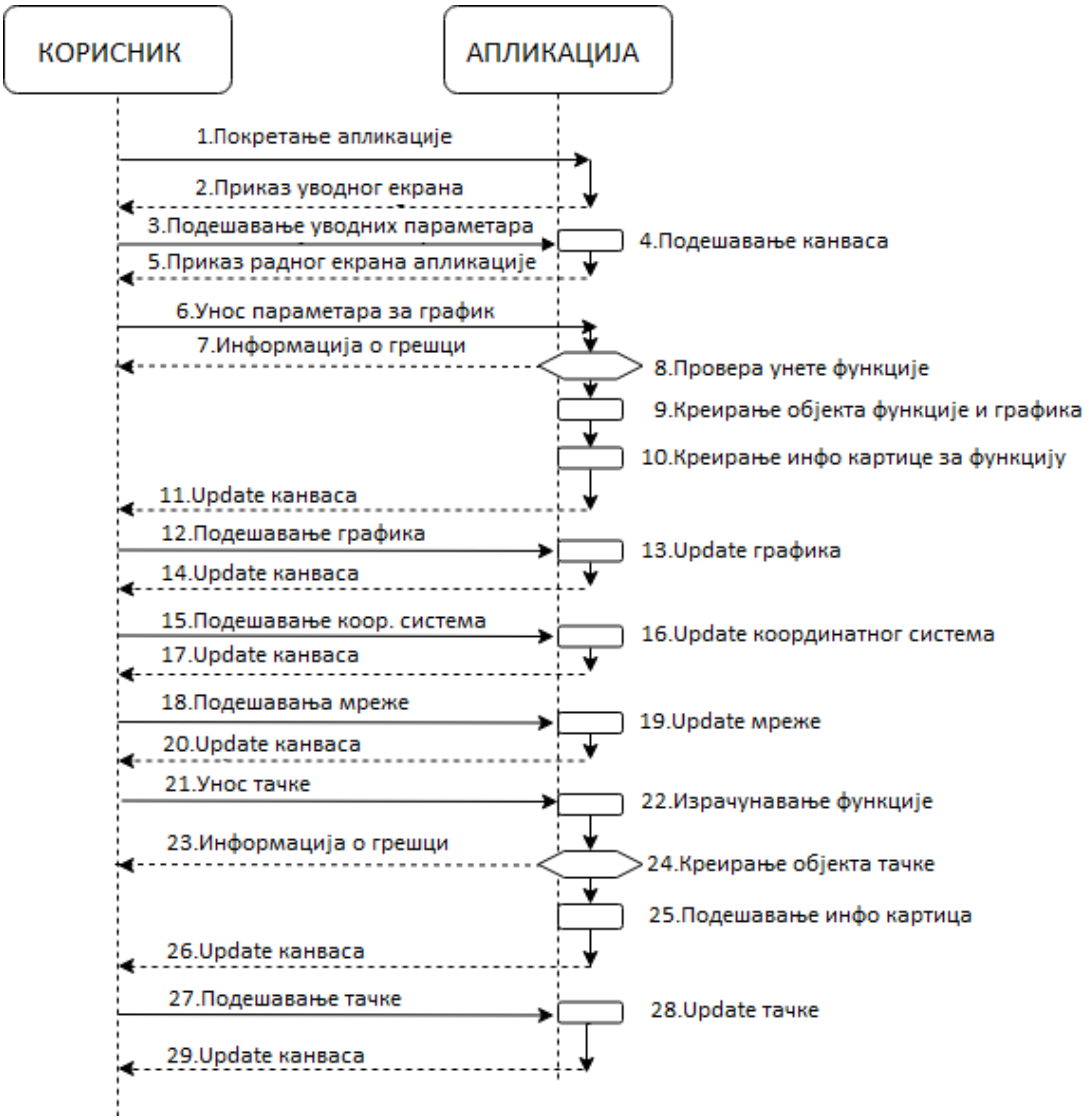
UML је стандардни језик за визуелно приказивање објектног модела у области софтверског инжењерства. Помоћу њега се моделују графички симболи потребни за приказивање жељеног система. За описивање овог пројекта су коришћени UML дијаграми: Use Case, Sequence, Activity, State machine и Class.

Дијаграм случајева (Use case) служи за визуелизацију понашања система, класа и интерфејса. Уједно специфира шта субјект ради, а не како ради. Типично се користи да специфира неку функционалност и понашање неког субјекта. Описује скуп секвенци акција, укључујући варијанте које субјекат обавља.



Слика 1. Use case дијаграм

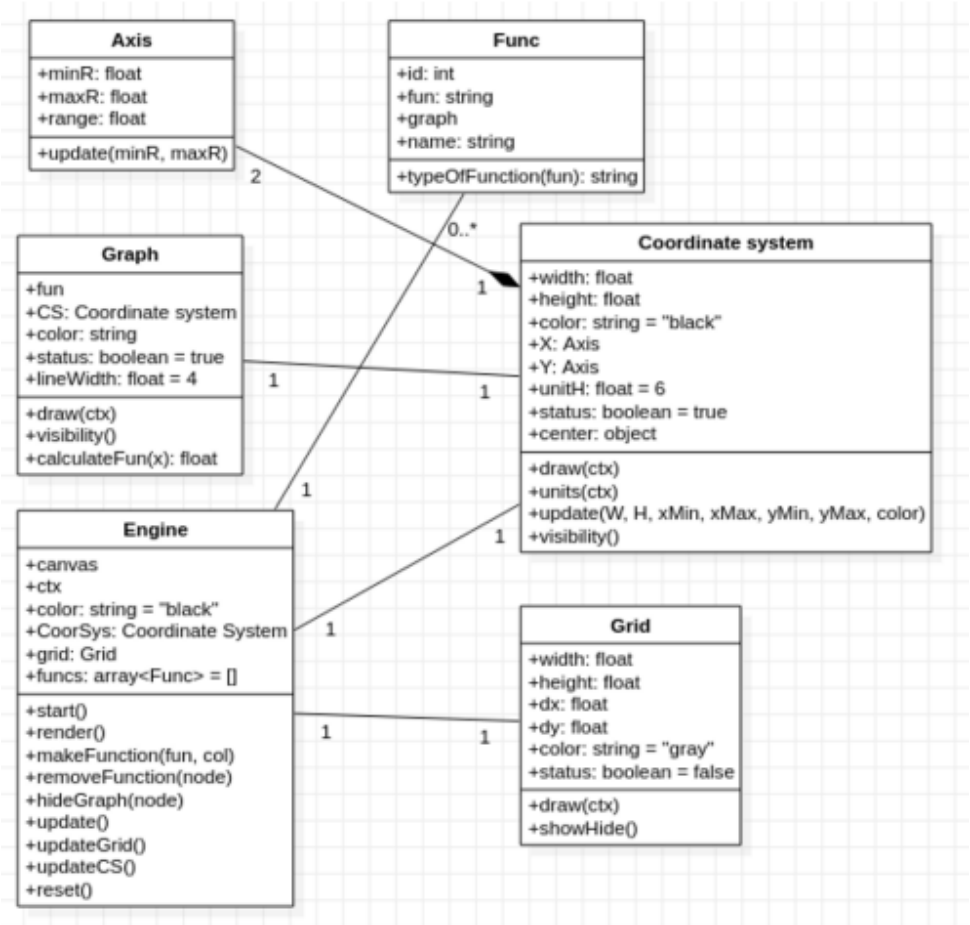
Дијаграм секвенци детаљно описује како се операције изводе, односно које поруке се шаљу и када. Користи се за приказ једног или више сценарија. Садржи две димензије: вертикалну (коришћена у овом пројекту) и хоризонталну.



Слика 2. Sequence дијаграм

State machine дијаграми се фокусирају на ток активности. Креирају се за ентитете који показују битно динамичко понашање.

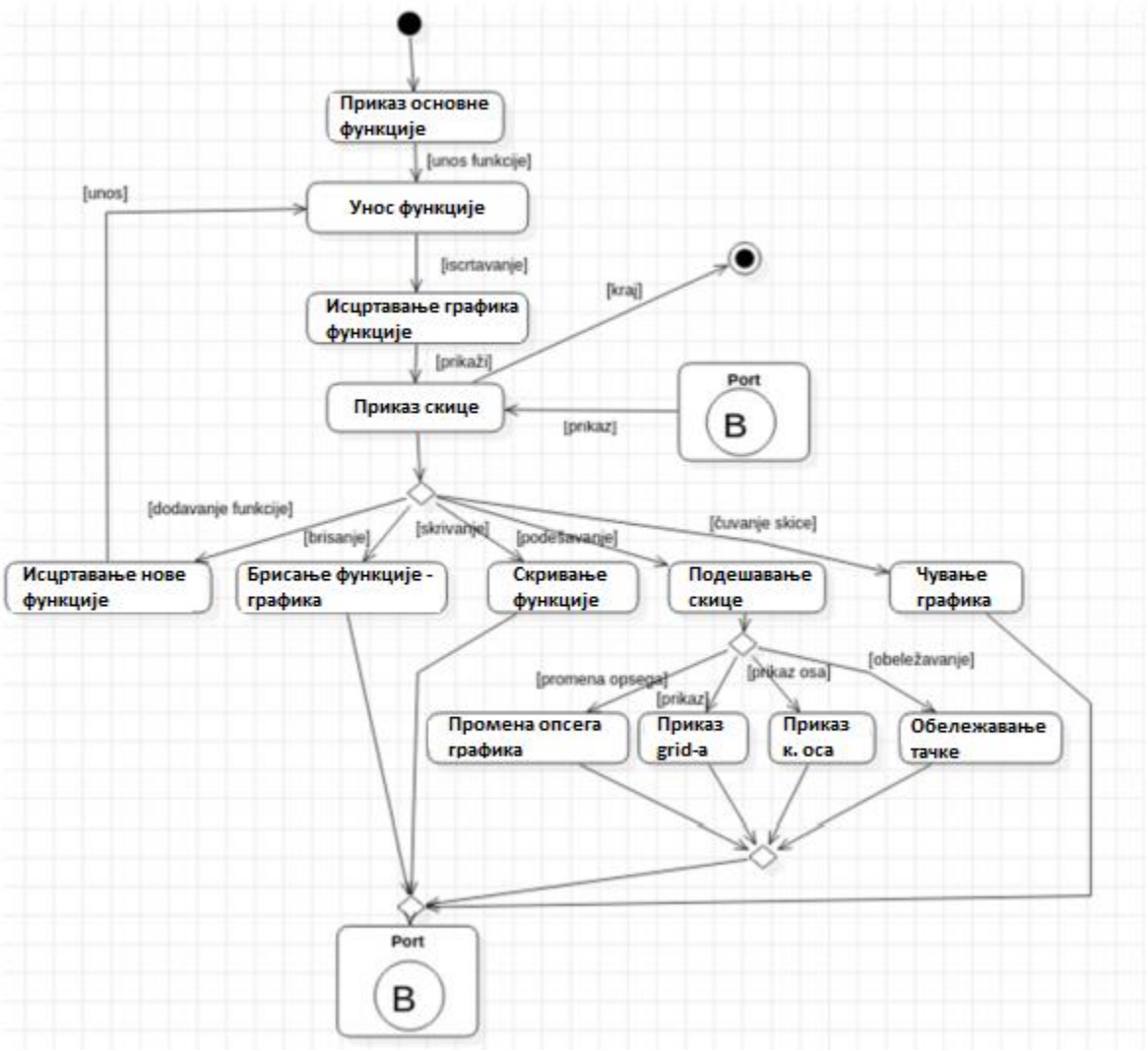
Слика 3. State machine дијаграм



Слика 4. Class дијаграм

Class дијаграм (Слика 4) приказује скуп класа, интерфејса, сарадњи и других ствари повезаних релацијама. Дијаграм класа је граф образован од темена (ствари) повезаних гранама (релацијама).

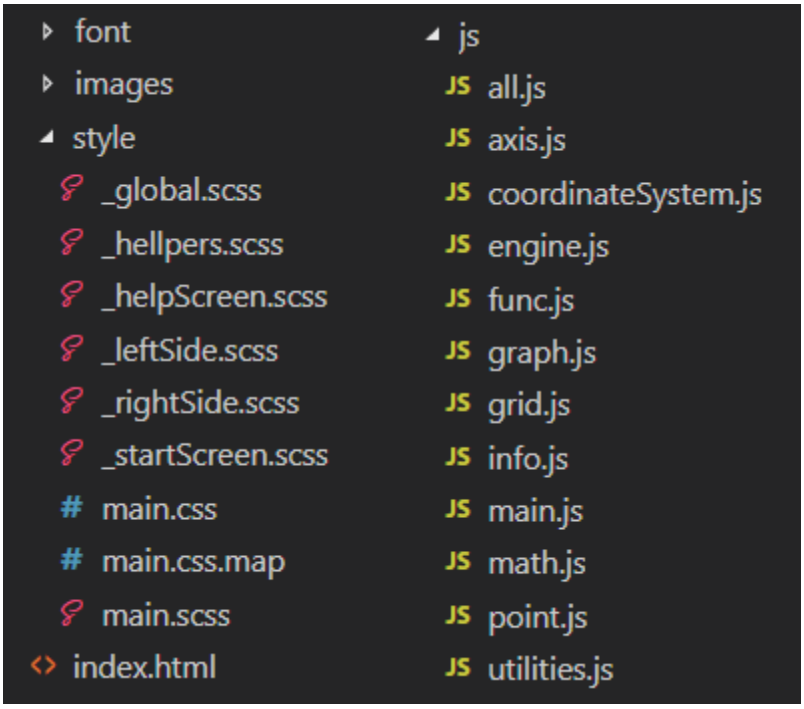
Activity дијаграми (Слика 5) служе за моделирање динамичких аспеката (понашања) система. Приказују ток активности извршавања објеката и ток објеката између корака активности.



Слика 5. Activity дијаграм

3. Структура апликације

Комплетна фунцкионалност апликације, која је предмет овог рада, развијена је помоћу JavaScript језика, за сам приказ у browser-у коришћен је HTML и наравно CSS за уређивање HTML кода. Као препроцесор за CSS коришћен је програмски језик SASS. На слици 6 представљена је структура фајлова унутар саме апликације.



Слика 6. Структура фајлова унутар апликације

Због релативно једноставног HTML кода целокупан HTML код садржан је у једном фајлу ***index.html*** . Овај фајл је заправо покретачки фајл апликације у browseru, уколико се апликација налази на серверу, при приступању url-у апликације кориснику се прво шаље овај фајл који у себи садржи све потребне информације о скриптама и стиловима које треба укључити.

У фолдеру **STYLE** налазе се ***.scss*** и ***.css*** фајлови апликације. Због мало боље прегледности само стилизовање апликације подељено је у шест различитих фајлова који се укључују у фајл **main.css** који се преводи у ***main.css*** који се даље укључује у ***index.html*** фајлу.

- **_global.scss –** садржи стилове за основне HTML елементе (body,div,ul,li...)
- **_hellpers.scss –** овај фајл садржи variable (boje,fontove,dimenzije) i miksine (функције) које се користе у осталим .scss фајловима.
- **_helpScreen.scss –** овај фајл садржи класе које уређују прозор упуства за коришћење апликације.

- **_leftSide.scss –** овај фајл садржи класе које уређују леву страну апликације, део за унос функција и део за чување информација о функцијама.
- **_rightSide.scss –** овај фајл садржи класе које уређују десну страну апликације, траку са алатима и сам канвас за исцртавање графика.
- **_startScreen.scss –** овај фајл садржи класе које уређују уводни прозор апликације

Битно је напоменути да се **.scss** користе само у развојној фази апликације, при коришћењу апликације клијенту се шаље само ***main.css*** фајл настао од ***main.scss*** фајла.

У фолдеру **IMAGES** налазе се све слике које се користе у апликацији, користи се укупно 14 које, скупа, заузимају приближно **3 MB.**

У фолдеру **FONTS** налазе се два фонта преузета са https://fonts.google.com/. Први фонт је **Montserrat** а други **Reckoner.**

Свакако најважнији део апликације јесу фајлови који се налазе у фолдеру **JS** овде се налазе скрипте које обезбеђују динамику HTML кода као и целокупну функционалност апликације. Главни фајл у овом фолдеру јесте ***main.js***, у овом фајлу налази се на десетине функција које имају улогу да манипулишу DOM-ом (**D**ocument **O**bject **M**odel) како би обезбедили потребну динамику web странице.

Као мера сигурности, целокупан JS код налази се унутар само позивајуће функције (Immediately Invoked Function Expressions - **IIFE**) тако да се не може приступити ниједној функцији која се налази изван опсега IIFE функције. JS код се покреће оног тренутка када се учита целокупан HTML и CSS код. Са слике 7 може се видети да се у самој апликацији прво креира објекат ***engine*** који представља инстанцу класе **Engine**. Затим се позивају функције за покретање апликације и додавање event-а на DOM елементима.

```
(function(){
let engine= new Engine();
window.onload = function(){
    setTimeout(intro,1500);
    events.click();
    events.onChange();
    events.submit();
    setMenuEvents();
    window.addEventListener('unload',()=>{…
    })
    window.addEventListener('reload',()=>{…
    })
}
```

Слика 7. Део кода из ***main.js*** фајла

Објекат event садржи три функције ***click, onChange*** и ***submit*** і свака функција у себи садржи више ***addEventListener*** функција које везују одређени евент за DOM елементе.

Све функције које се налазе унутар ***main.js*** фајла подељене су у следеће групе:

- Функције за иницирање event-a слика 8.а)
- Функције за стартовање апликације и стартна подешавања слика 8.б)
- Функције за контролу инпута слика 8.ц)
- Функције за динамичко додавање DOM елемената 8.д)
- Функције за контролу алата апликације слика 8.е)

```
35  let events = {
36      click: function(){ ···
107     },
108     onChange:function(){ ···
122     },
123     submit:function(){ ···
141     }
142 }
```

Слика 9.а) Event функције

```
145 function intro(){ ···
165 }
166 function setEngine(e){ ···
208 }
209 function renderApp(h){ ···
219 }
220 function infoDefault(){ ···
222 }
223 function initialSettings(){ ···
226 }
227 function setDefaultSettings(){ ···
238 }
239 function showCustomColor(e){ ···
248 }
249 function showHideHelp(){ ···
251 }
```

Слика 9.б) Стартне функције

```
256 function changeIconColor(e,color){ ···
264 }
265 function openFunctionInput(){ ···
267 }
268 function loadInput(){ ···
284 }
285 function resetInput(){ ···
298 }
299 function insertInput(e){ ···
309 }
```

Слика 9.ц) Функције за контролу инпута

```
312 function addCard(f){ ···
350 }
351 function addCardEvents(el){ ···
386 }
387 function removeCard(id,el){ ···
391 }
392
```

Слика 9.д) Функције за додавање DOM елемената

```
395 function setCsRange(e){ ···
405 }
406 function setMenuEvents(e){ ···
446 }
447 function setRangeInput(){ ···
464 }
465 function setPinPointMenu(el){ ···
482 }
483 function checkPinPointInput(form){ ···
517 }
518 function cardAddPoint(id,xp,yp){ ···
548 }
549 function addPointEvents(info,id){ ···
569 }
570 function setChanges(el){ ···
635 }
636 function saveFile(e){ ···
646 }
647 function loadFile(e){ ···
666 }
```

Слика 9.е) функције за контролу алата апликације

Фајлови ***all.js*** и ***math.js*** нису писани за ову апликацију већ су преузети из „open source“ извора за JS. Библиотека ***all.js*** садржи код потребан за коришћење **SVG** графике, ова библиотека је преузета са сајта https://fontawesome.com/ који су и развили поменуту библиотеку **SVG** графике. Фајл ***math.js*** представља библиотеку која садржи различите математичке функције. У овом пројекту користи се само једна фунцкија из ове библиотеке и то је функција за евалуацију стринг као математичког израза. Стринг који се покупи као математичка вредност функције, коју корисник апликације унесе, процесуира се преко функције ***eval()*** која преузима стринг а враћа бројну променљиву, више информација о самој библиотеци могуће је видети на http://mathjs.org/. Фајл ***utilities.js*** садржи неколико помоћних функција које се употребљавају у више наврата.

Остатак фајлова унутар фолдера JS представља класе које се користе приликом исцртавања графика функције и манипулације истим. Главна класа која контролише целу ОО структуру јесте **Engine** стога највише пажње посветиће се баш овој класи.

Приликом инцирања објекта класе **Engine** покреће се конструктор функција класе. Ова функција сетује низ за функције и пар објеката за складиштење података о самом стању у коме се апликација налази. Прва метода која се позива од стране новонасталог објекта јесте ***start*** (слика 10). Ова метода креира канвас и два објекта координатни систем (***CoorSys***) и мрежу (***grid***) на основу добијених вредности. Након креирања поменутих објеката покреће се метода ***render*** која има улогу да исцртава потребну графику на канвасу.

```
20      start(w,h,xMin,xMax,yMin,yMax,csColor,canColor,gCol){
21          this.setCanvas(w,h,canColor)
22          this.CoorSys = new CoordinateSystem(w,h,xMin,xMax,yMin,yMax,csColor,10);
23          this.grid    = new Grid(w,h,10,10,csColor);
24
25          this.render();
26          this.input.color = gCol;
27      }
```

Слика 10. Метода старт у класи Engine

Класа **CoordinateSystem** креира објекат који представља координатни систем апликације. Објекат креиран уз помоћ ове класе садржи неколико метода које се позивају из класе **Engine**. Ове методе служе за графички приказ координатног система на канвасу као и за подешавање различитих параметара објекта.

Класа **Grid** креира објекат који представља мрежу апликације. Ова мрежа је невидљива приликом старта апликације и може се учинити видљивом позивањем методе ***showHide*** поред ове методе објекат креиран уз помоћ ове класе садржи методе за исцртавање саме мреже на канвасу као и методе за манипулацију самим изгледом мреже.

Да се настави даље са класом **Engine**, као што је поменуто, унутар методе ***start*** позива се метода ***render*** која која исцртава целокупан садржај канваса. Ова метода провера тренутни статус видљивости сваког објекта који треба графички представити и у колико је видљив позива ***draw*** функцију на том објекту.

На слици 11 је приказана структура методе ***render.***

```
45      render(){
46          this.ctx.clearRect(0,0,this.canvas.width,this.canvas.height);
47          this.ctx.save()
48          if(this.grid.status){
49              this.grid.draw(this.ctx);
50          }
51          if(this.CoorSys.status){
52              this.CoorSys.draw(this.ctx);
53          }
54          for(let i=0;i<this.funcs.length;i++){
55              let points = this.funcs[i].points;
56              if(this.funcs[i].graph.status){
57                  this.funcs[i].graph.draw(this.ctx)
58              }
59              for(let j =0 ; j<points.length;j++){
60                  if(points[j].status){
61                      points[j].draw(this.ctx)
62                  }
63              }
64          }
65          this.ctx.restore();
66
67      }
```

Слика 11. Структура методе render у класи Engine

Остале методе унутар класе Engine покрећу се приликом интеракције корисника са самом апликацијом. Метода ***makeFunction*** (слика 12) покреће се пошто корисник унесе потребне информације за исцртавање графика функције и притисне дугме за исцртавање графика (погледај поглавље 4 за више информација). Пре покретања поменуте методе врши се провера уноса корисника преко методе ***checkInput*** из фајла **main.js** у колико израз функције није валидан метода ***makeFunction*** се не покреће. Сама метода се покреће из функције ***addCard*** из фајла **main.js**. Улога ове методе јесте да направи нов објекат **Func** и складишти га у низ ***funcs*** који представља променљиву унутар објекта класе **Engine**.

```
69      makeFunction(){
70          this.funcs.push(new Func(this.input.value,this.input.color,this.CoorSys,this.input.width))
71          this.update();
72
73      }
```

Слика 12. Структура методе ***makeFunction***

Класа **Func** креира објекат који садржи све информације о унетој функцији (ид, вредност, боја графика итд). Приликом креирања објекта класе **Func** у конструктору се аутоматски креира објекат класе **Graph** који садржи све информације везане за график функције као и методе потребне за исцртавање графика на канвасу. На слици 13 приказана је структура класе **Func**.

```
class Func {
    constructor(val,col,cs,lw){
        this.id     = uniqueId();
        this.value  = val;
        this.width  = lw;
        this.color  = col;
        this.cs     = cs;
        this.graph  = new Graph(this.value,cs,col,lw);
        this.name   = this.typeOfFunction(this.value);
        this.points = [];
    }

    typeOfFunction(val){ ···
    }
    addPoint(ctx,x,y,col="black",rad=2){ ···
    }
    removePoints(){ ···
    }
    hidePoints(){ ···
    }
}
```

Слика 13. Структура класе **Func**

Као што се може видети са слике 13, поред конструктора, класа func садржи још четири методе. Метода ***typeOfFunction*** се покреће директно из конструктора и има улогу да одреди тип функције на основу унете вредности функције, ова метода враћа стринг који представља тип функције (Линеарна, Квадратна, Тригонометријска итд). Метода **addPoint** креира нов објекат класе **Point** који се складишти у низ ***points***, који представља променљиву у објекту класе **Func**. Метода ***removePoints*** уклања објекат класе **Point** из низа ***points***. Метода ***hidePoints*** подешава видљивост тачке на канвасу.

Класа **Point** се налази у фајлу ***point.js*** и има улогу да инстанцира објекат који представља тачку. Овај објекат садржи променљиве ***X*** и ***Y*** које представљају координате на канвасу променљиву ***status*** која дефинише видљивост тачке, променљива ***color*** која дефинише боју тачке и променљива ***radius*** која дефинише пречник тачке. Поред тога ова класа садржи неколико метода за исцртавање тачке и манипулисање тачком.

Да се опет вратимо на методу ***makeFunction*** из класе **Engine**. Пошто се направи нов објекат класе **Func** позива се метода ***update*** ова метода заправо само позива методу ***render*** како би се целокупна графика на канвасу поново исцртала. Може се позивати и директно функција ***render*** у исте сврхе.

Метода ***removeFunction*** из класе **Engine**, позива се када корисник кликне на дугме које има улогу да избрише функцију и њен графички приказ на канвасу. На слици 14 приказана је структура ове методе која се покреће из методе ***removeCard*** из фајла ***main.js***.

```
74      removeFunction(id){
75          for(var i=0;i<this.funcs.length;i++){
76              if(this.funcs[i].id == id){
77                  var ind = this.funcs.indexOf(this.funcs[i])
78                  this.funcs.splice(ind,1)
79              }
80          }
81          this.update();
82      }
```

Слика 14. Структура методе ***removeFunction*** из класе **Engine**

Методе ***setPoint***, ***hidePoint*** и ***deletePoint*** (слика 15), из класе **Engine**, покрећу се преко функција из фајла ***main.js.*** Метода ***setPoint*** креира нов **Point** објекат који се смешта у одговарајући **Func** објекат, метода ***hidePoint*** подешава видљивост **Point** објекат, почетна вредност јесте да објекат буде видљив приликом графичког приказа на канвасу. Метода ***deletePoint*** уклања жељену тачку из низа ***points*** који представља променљиву у класи **Func**.

```
84      setPoint(x,y,col,rad,id){
85          this.funcs[id].addPoint(this.ctx,x,y,col,rad)
86          this.update();
87      }
88      deletePoint(fId,pId){
89          let pArr   = this.funcs[fId].points,
90              j=0,ind;
91              pArr.forEach(p=>{
92                  if(p.id == pId){
93                      ind = j;
94                  }
95                  j++;
96              })
97
98             this.funcs[fId].points.splice(ind,1);
99          this.update();
100     }
```

Слика 15. Структура метода ***setPoint*** и ***deletePoint*** из класе **Engine**

У току рада у апликацији могуће је променити поједине параметре: канваса, координатног система, мреже и самог графика функције. Сви параметри се мењају позивањем одговарајуће методе из **Engine** класе. Методе се активирају клик евентом на одговарајући HTML елемент, на крају сваке методе позива се ***update*** метода из класе **Engine** како би се графички приказ на канвасу ажурирао.

Што се тиче самог канваса из саме апликације могуће је променити само боју позадине канваса, ово се ради уз помоћ методе ***updateCanvasCol***.

За подешавање графика постоје две методе (слика 16), прва је ***updateGraphVisibility*** која подешава видљивост графика функције, друга је ***updateGraphColor*** и подешава боју графика.

```
160     updateGraphVisibility(id){
161         for(var i=0; i<this.funcs.length;i++){
162             if(this.funcs[i].id == id){
163                 this.funcs[i].graph.visibility();
164             }
165         }
166         this.update()
167     }
168     updateGraphColor(col,id){
169         this.funcs.forEach(f=>{
170             if(f.id == id){
171                 f.graph.update(col);
172             }
173         })
174         this.update()
175     }
```

Слика 15. Структура метода за подешавање графика функције

Постоје три методе за подешавање **Grid** објекта тј. мреже канваса (слика 16). Прва метода је ***updateGridVisibility*** и подешава видљивост мреже. Друга метода је ***updateGridColor*** и подешава боју мреже. Последња метода је ***updateGridDimension*** ***updateGridDimension*** уз помоћ ове методе одређује се број ћелија мреже.

```
119     updateGridVisibility(){
120         this.grid.showHide();
121         this.update();
122     }
123     updateGridColor(col){
124         this.grid.updateColor(col);
125         this.update();
126     }
127     updateGridDimension(w,h,ux,uy){
128         this.grid.updateDimension(w,h,ux,uy);
129         this.update();
130     }
```

Слика 16. Структура метода за подешавање мреже

За подешавање координатног система постоји шест метода у класи **Engine** и свака позива одговарајућу методу објекта ***CoorSys*** из класе Енгине. На слици 17 приказана је структура поменутих метода.

```
132     updateCS_visibility(){
133         this.CoorSys.visibility();
134         this.update();
135     }
136     updateCS_range(...args){
137         this.CoorSys.updateRange(...args);
138         this.update();
139     }
140     updateCS_text(num){
141         this.CoorSys.updateTextSize(num);
142         this.update();
143     }
144     updateCS_uh(num){
145         this.CoorSys.updateUnitSize(num);
146         this.update();
147     }
148     updateCS_col(col){
149         this.CoorSys.updateColor(col);
150         this.update();
151     }
152     updateCS_textVis(){
153         this.CoorSys.showTextToggle()
154         this.update();
155     }
```

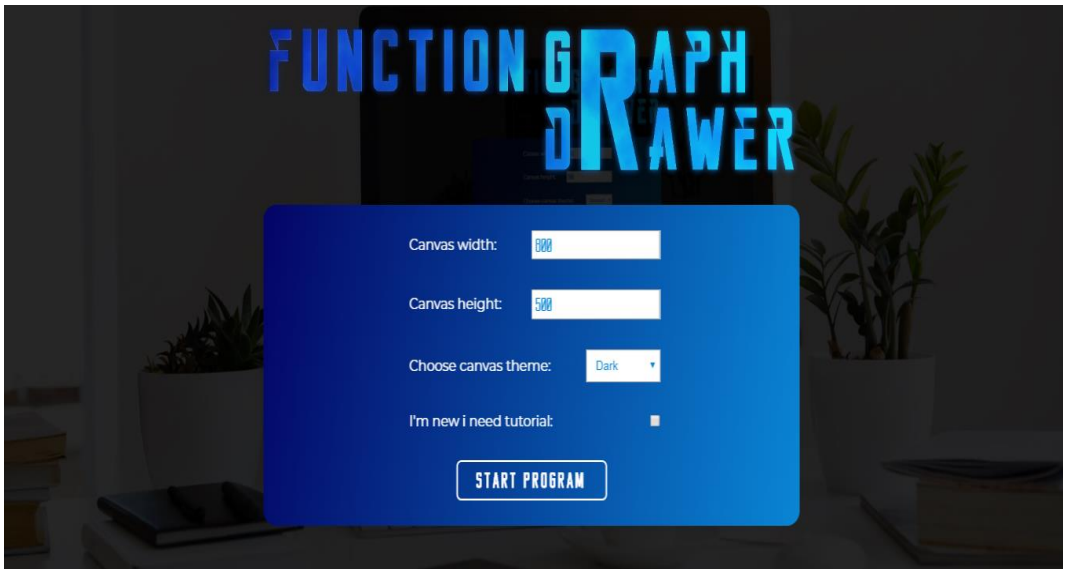
Слика 16. Структура метода за подешавање координатног система

Метода ***updateCS_visibility*** подешава видљивост координатног система. Метода ***updateCS_range*** подешава опсег оса координатног система. Метода ***updateCS_text*** подешава величину фонта који се користи за приказ мерних јединица на координатним осама. Метода ***updateCS_uh*** подешава висину подеоних елемената на координатном систему. Метода ***updateCS_col*** подешава боју координатног система. И за крај метода ***updateCS_textVis*** подешава видљивост текста који представља вредност мерних јединица координатног система.

4. UI(User Interface) апликације и опис рада

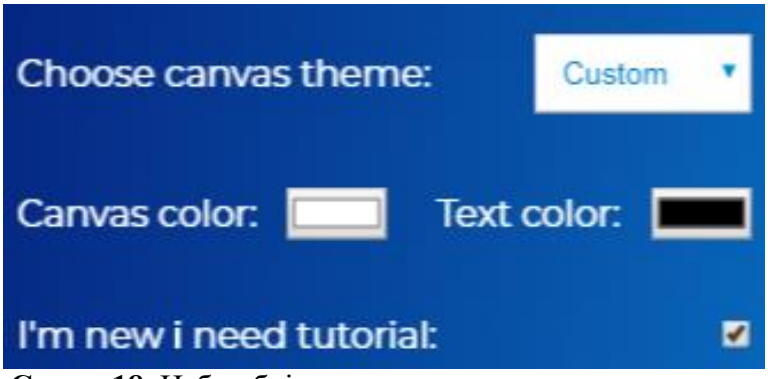
4.1. Уводни екран апликације

При покретању апликације корисник у се прво приказује уводни екран (слика 17). Уводни екран садржи два текст инпута за унос димензија канваса. Аутоматски се подешавају препоручене вредности за тип уређаја на коме се корисник налази. После подешавања димензија канваса могуће је подесити тзв. тему канваса. Ова подешавања директно утичу на боју канваса и боју координатног система канваса.



Слика 17. Уводни екран апликације

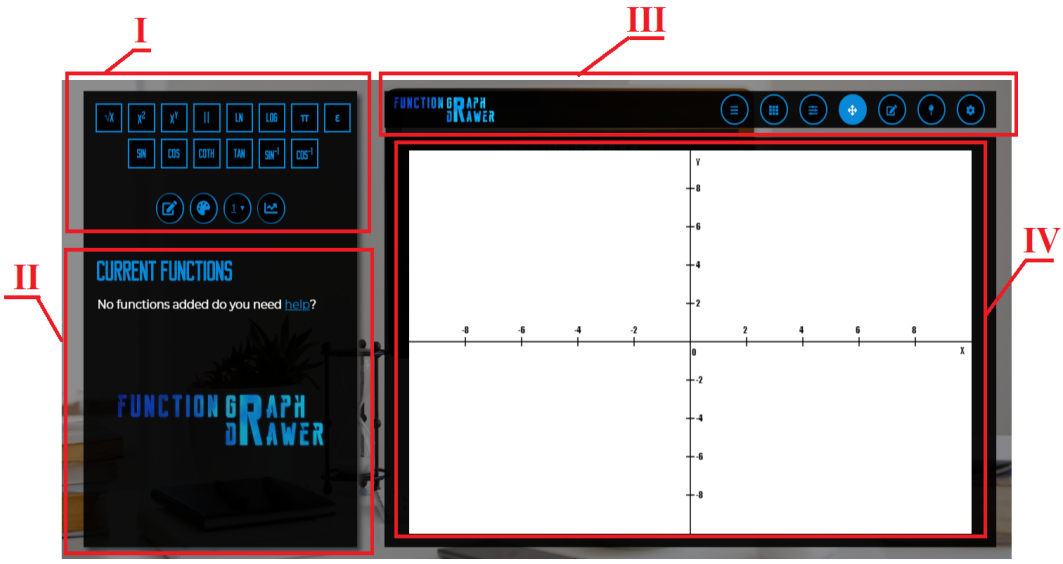
Из подешавања теме могуће је изабрати четири основне опције **STANDARD** (канвас беле боје координатни систем црне), **LIGHT** (канвас беле боје координатни систем плаве), **DARK** (канвас црне боје координатни систем плаве) и **CUSTOM** (корисник може да бира вредности за боје канваса и координатног система (слика 18). Чекирањем на туториал приликом стартовања апликације отвара се прозор са упуствима за коришћење апликације.



Слика 18. Избор боје канваса и координатног система

По притиску на дугме start програм покреће се главни екран апликације. Уколико је опција за tutorial чекирана аутоматски се отвара прозор са упуство за коришћење апликације. Када се апликација покрене корисник види поставку приказану на слици 19. Сам интерфејс апликације може се поделити у четири главна сегмента (слика 19):

- I сегмент - део за унос функције
- II сегмент - део за информације о унетим функцијама
- III сегмент - део са са алаткама за контролу канваса и саме апликације
- IV сегмент - канвас за исцртавање графика



Слика 19. UI апликације подељен у четири главна дела

4.2. I сегмент – унос функције

Овај део апликације са налази у горњем левом углу UI-а, његова улога је да омогући кориснику да унесе зељену функцију. Горњи део првог сегмента (слика 20) састоји се од дугмића за брз унос неких честих математичких функција и операција, попут корена, степеновања, тригонометријских и логаритамских функција итд.

Слика 20. I сегмент апликације

У доњем делу првог сегмента налази се инпут за унос самог израза функције као и додатна подешавања за боју графика и дебљину линије графика. Овај део задржи четири дугмета (слика 21). Притиском на прво дугме отвара се инпутни део за унос функције, он се такође отвара притиском на било које дугме из горњег дела I сегмента. Друго дугме омогућава кориснику да изабере боју за график функције, у колико се боја не изабере бице аутоматски изабрана, и то боја која је комплементарна боји канваса како би се график видео на канвасу, боја може бити промењена у било ком тренутку. Треће дугме подешава дебљину линије графика. Четврто дугме исцртава график унете функције (у колико је унета исправна функција).

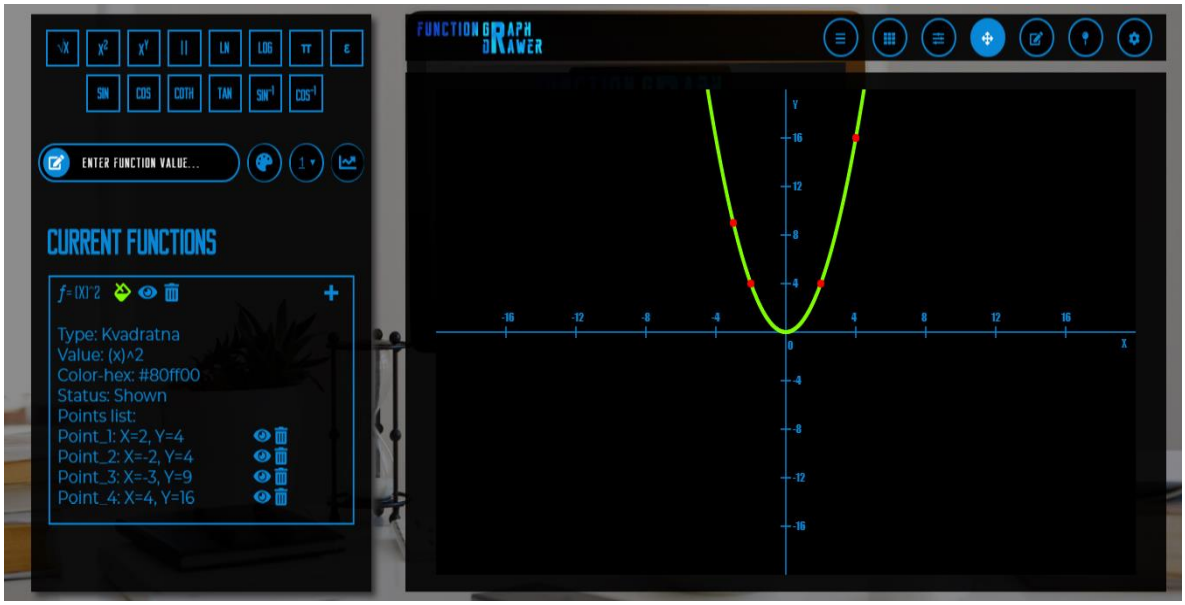
Слика 21. Алатке за унос вредности функције

4.3. II сегмент – информације о унетим функцијама

Приликом креирања функције и исцртавања њеног графика све информације складиште се у доњем левом углу апликације. Свака функција добија своју картицу која садржи основне информације о функцији као и основне команде за манипулацију са самим графиком. На примеру са слике 22 унете су три различите функције и креирана је картица за сваку функцију понаособ. Гледајући са лева на десно, свака картица на почетку има информацију о самом изразу функције, ова информација не може бити промењена. Затим, свака картица има 3 дугмета, прво дугме омогућава промену боје графика тренутне функције, друго дугме омогућава кориснику да сакрије график те функције, треће дугме брише график функције као и саму картицу (функција не може бити поново враћена). На крају картице налази се дугме за отварање падајућег менија картице (слика 22).

Слика 22. Картице са информација о функцијама

Падајући мени картице садржи информације о самој функцији као и скуп тачака са кординатама X и Y које су накнадно унете за ту функцију о чему ће бити речи касније. Поменуте тачке се могу обрисати и сакрити из овог падајућег менија. Ово се може видети са примера на слици 23, где су за квадратну функцију унете 4 тачке и сваку посебно је могуће избрисати или сакрити.



Слика 23. Пример графика квадратне функције са активираним падајућим менијем картице

4.4. III сегмент – трака са алатима

Овај сегмент UI-а позициониран је изнад самог канваса и састоји се од седам дугмади, свако има своју посебну функционалност за контролу канваса као и саме апликације. У даљем тексту свако дугме позиваће се на основу ознака са слике 24.

Слика 24. Трака са алатима за контролу апликације

Дугме 1 отвара падајући мени који кориснику пружа шест различитих опција:

• **NEW** – апликација се враћа на уводни екран и сав тренутни рад је изгубљен

• **RESET** – сви графици функција се неповратно бришу као и све картице које садрже информације о функцијама. Задржавају се тренутна подешавања канваса, координатног система и мреже.

• **SAVE** – чува тренутни рад у локалној меморији

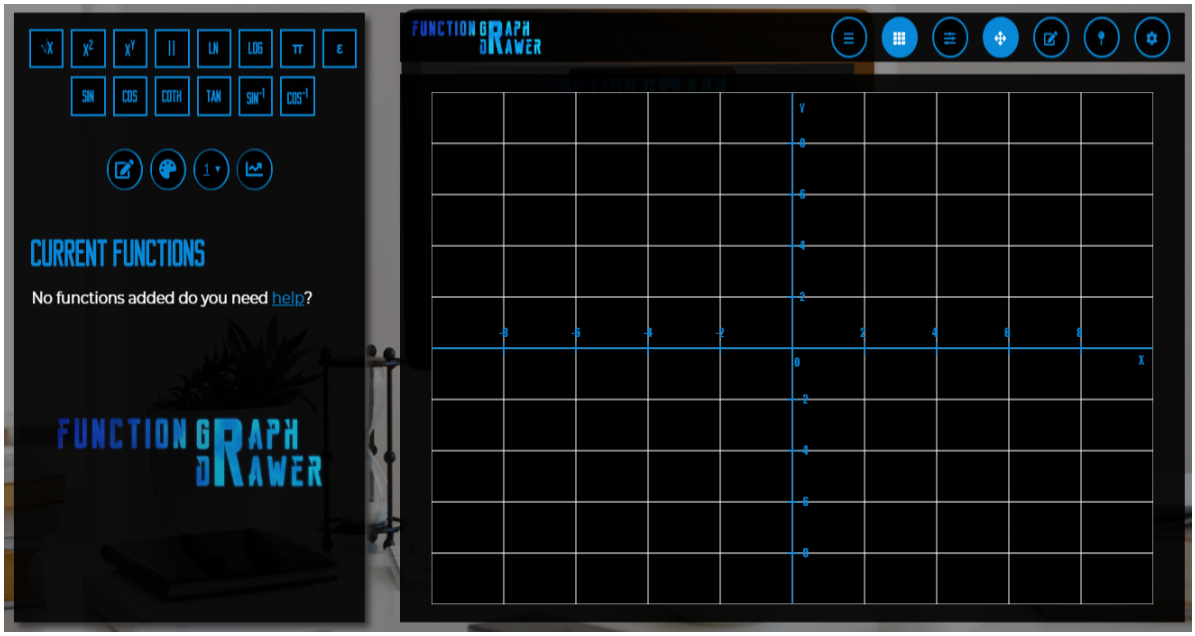
• **LOAD** – учитава неки од претходних радова из локалне меморије

• **HELP** – отвара упуство за коришћење апликације

• **QUIT** – затвара падајући мени

Треба напоменути да опције саве и лоад раде са **LOCAL STORAGE**-ом и све информације о тренутном радном прогресу чувају се **JSON** фајлу. У општим подешавањима апликације (биће речи касније) могуће је избрисати све снимљене податке.

Дугме 2 је тзв. кликабилно дугме, има два стања. Прво стање (активно при покретању апликације) сакрива мрежу на канвасу. Друго стање приказује мрежу на канвасу (слика 25). Изглед мреже (број ћелија и боју) могуће је подешавати унутар општих подешавања апликације.

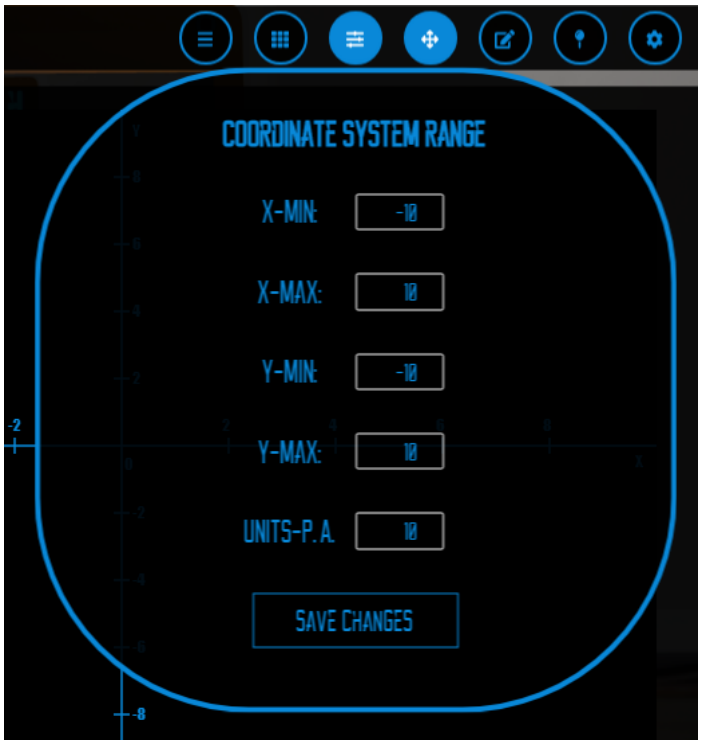


Слика 25. Апликација са активираном мрежом

Дугме 3 игра веома важну улогу у ефикасном коришћењу апликације. Наиме, притиском на ово дугме отвара се нов прозор који кориснику даје могућност да подешава опсег координатних оса. Стандардна подешавања постављају X и Y осе да иду у опсегу од -10 до 10. Наравно овај опсег може одговарати при исцртавању графика појединих функција али исто тако може бити непрактичан при исцртавању графика неких других функција. Тренутно максимални опсег оса координатног система јесте од -100.000 до 100.000.

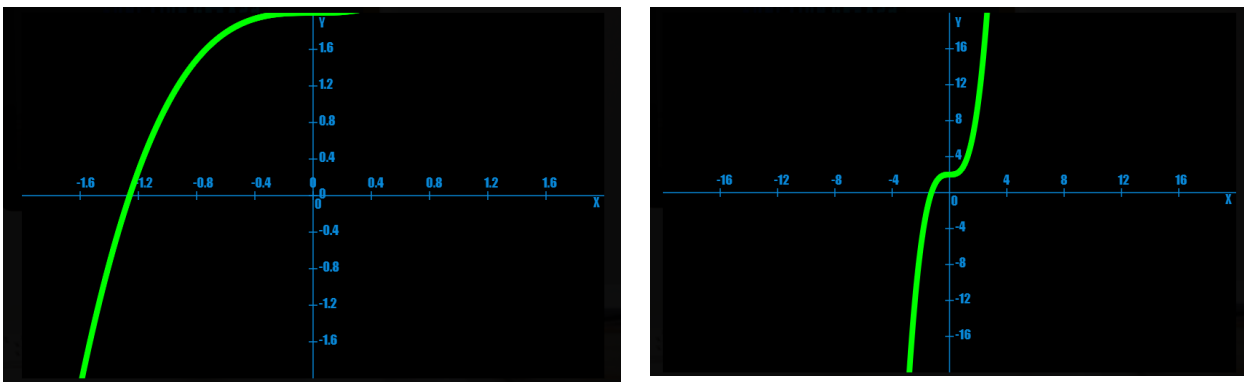
У прозору који се отвара притиском на дугме 3 могуће је подесити и број подеоних јединица по X и Y оси. Овде треба напоменути да се сама вредност ових јединица заокружује на две децимале иза нуле (у колико се ради о бројевима са децималним остатком). Стога, није пожељно подешавати опсег оса да иде испод -0.01 и 0.01.

На слици 26 приказан је прозор који се приказује притиском на дугме 3. Овде се може видети да су за све осе већ унете вредности, ове вредности представљају тренутне вредности опсега координатног система.



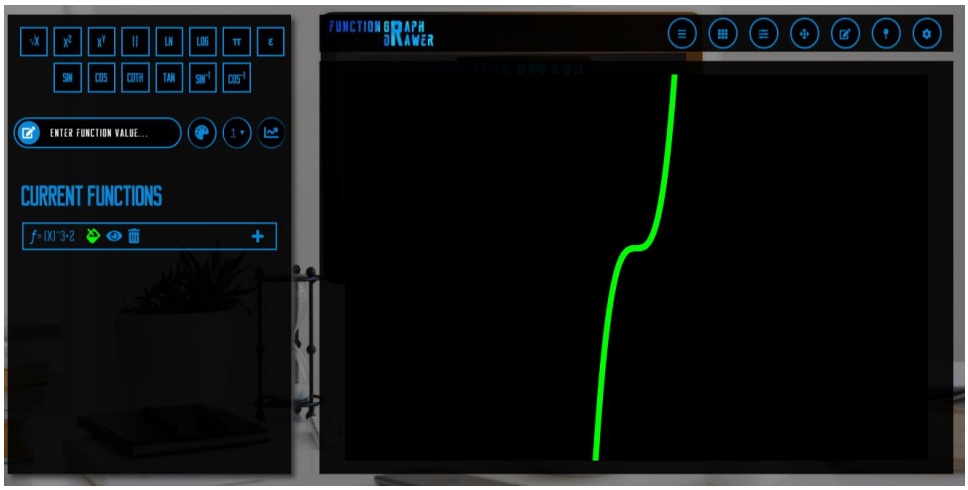
Слика 26. Прозор за подешавање координатног система

Како би се стекао бољи увид у важност опсега координатног система иста функција $Y=X^3+2$ приказана је у опсегу од -2 до 2 (слика 27.а) и у опсегу од -20 до 20 (слика 27.б).



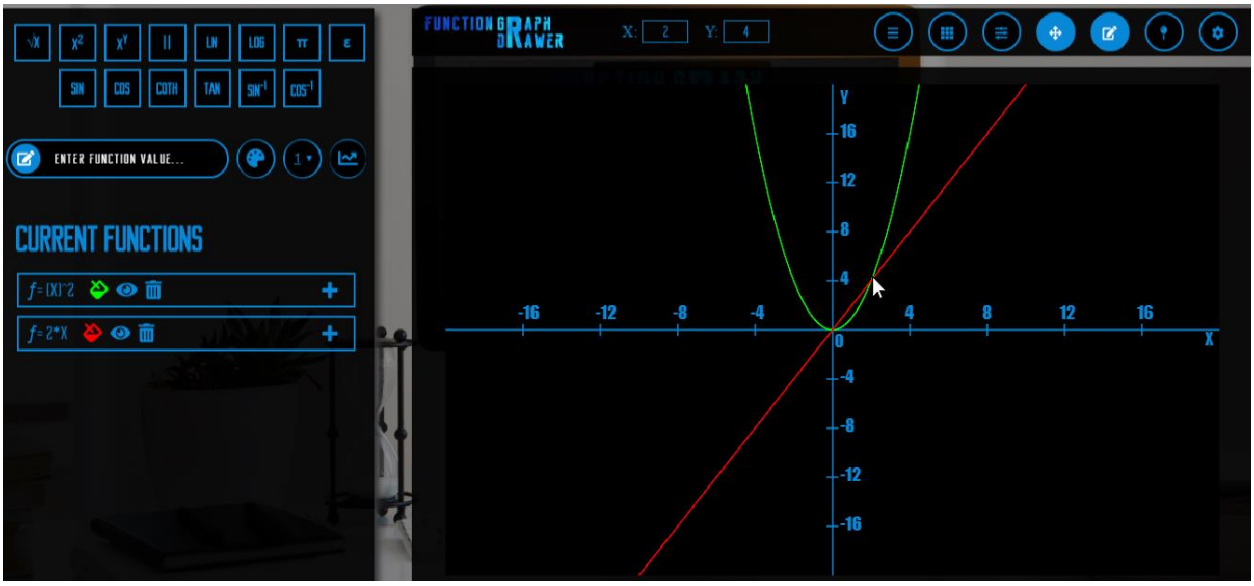
Слика 27. Функција **Y=X3+2; а**) у опсегу од -2 до 2**; б)** у опсегу од -20 до 20

Дугме 4 је такође кликабилно дугме са два стања, његова улога је да контролише видљивост координатног система. При покретању апликације ово дугме се налази у активном стању и координатни систем је приказан. На слици 28 приказана је функција са слике 27б) са угашеним координатним системом.



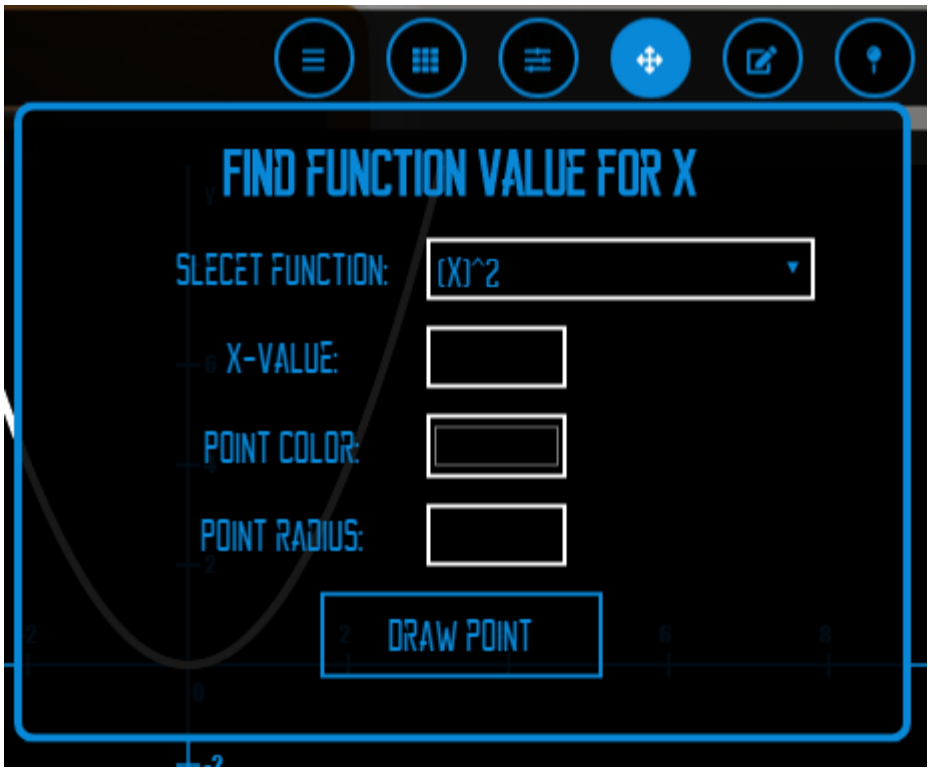
Слика 28. Изглед канваса са сакривеним координатним системом

Дугме 5 је још једно кликабилно дугме са два стања, оно контролише видљивост output-а за координате X и Y на канвасу које се генеришу на основу тренутног положаја курсора миша на канвасу. Ове вредности се такође заокружују на две децимале, ово је корисно за добијање координата било које тачке у кординатном систему, може бити прилично корисна када је потребно одредити тачке пресека на графицима и слично. На слици 29 је приказано како је лако одредити координате тачке пресека две функције X^2 и $2*X$.



Слика 29. Активно дугме 5 са курсором унутар канваса на месту пресека две функције

Дугме 6 отвара нов прозор који даје могућност провере вредности функције за унету вредност променљиве X као и исцртавање тачке на координатама унетог X и израчунатог Y (у колико се тачка налази у тренутном домену координатног система). На слици 30 приказан је прозор за унос података за додавање поменуте тачке.



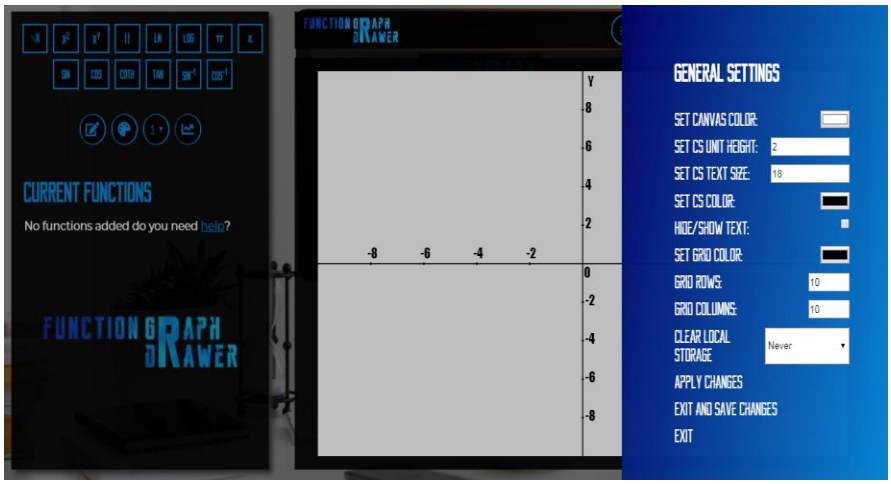
Слика 30. Прозор за проналажење вредности изабране функције за унето X и исцртавање тачке на тим координатама

На слици 30 може се видети прозор који настаје притиском на дугме 6, у овом прозору корисник треба да унесе 4 параметра. Први параметар (почевши од врха ка дну) представља избор функције за коју корисник жели да израчуна вредност, други параметар представља вредност за X, трећи и четврти параметар нису обавезни служе за одређивање боје тачке и њене величине. Све информације о унетим тачкама могу се наћи у падајућем менију картица које садрже информације о одређеним функцијама.

Дугме 7 је последње дугме на траци са алатима, оно представља дугме за отварање општих подешавања апликације. У општим подешавањима налази се више подешавања везаних за сам канвас и његов приказ на екрану. У оквиру ових подешавања постоје следеће опције:

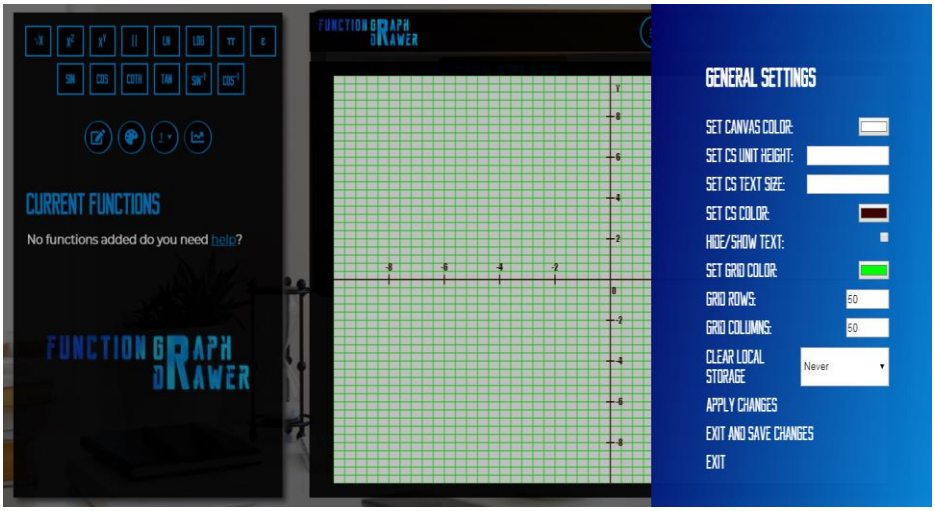
- **SET CANVAS COLOR** – ова опција кориснику омогућава да подеси боју канваса.
- **SET CS UNIT HEIGHT** – ова опција омогућава кориснику да подеси висину подеоних јединица на кординатном систему (слика 31).

- **SET CS TEXT SIZE** – ова опција омогућава кориснику да подеси величину текста који представља вредности подеоних јединица координатног система (слика 31).



Слика 31. Подешавање величине текста и подеоних јединица координатног система

- **SET CS COLOR –** ова опција омогућава корисниику да подеси боју координатног система.
- **HIDE/SHOW TEXT –** ова опција омогућава кориснику да сакрије вредности подеоних јединица на координатном систему.
- **SET GRID COLOR –** ова опција служи за подешавање боје мреже (слика 32).
- **GRID ROWS –** ова опција служи за подешавање хоризонталног броја ћелија мреже (слика 24).
- **GRID COLUMNS –** ова опција служи за подешавање вертикалног броја ћелија мреже (слика 24).

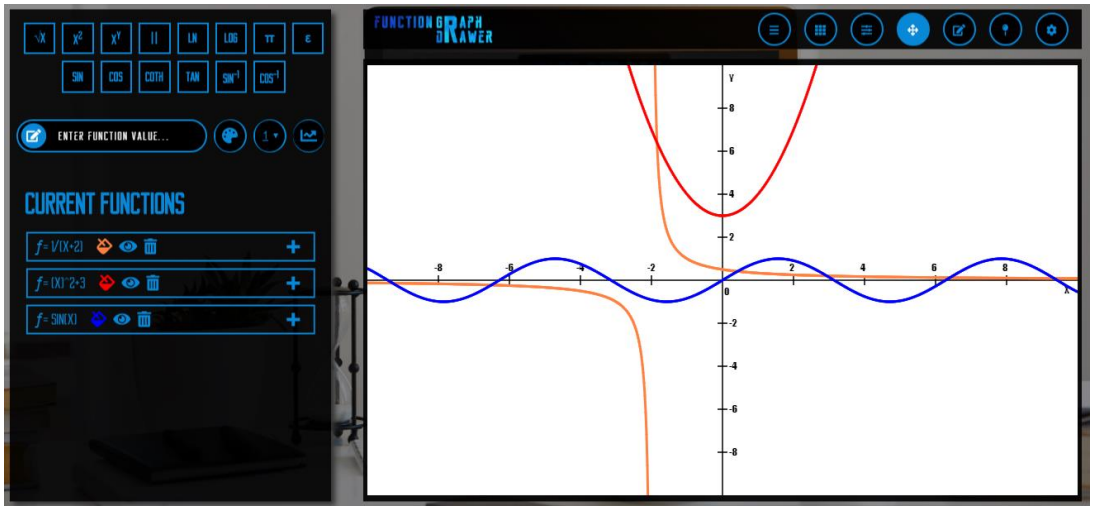


Слика 32. Подешавање броја ћелија мреже и боје мреже

- **CLEAR LOCAL STORAGE** – ова опција омогућава контролу локалне меморије клијента за апликацију (меморије у коју се уписују снимљени пројекти). Могуће је изабрати четири опције: **NEVER**, **ON PAGE RELOAD**, **ON BROWSER CLOSED**, **CLEAR NOW.** При покретању ова опција је постављена на **NEVER** што значи да се подаци не бришу из локалне меморије (податке може обрисати неки други софтвер или сам корисник). Опција **ON PAGE RELOAD** брише све податке из лоцал стораге-а при сваком поновном учитавању странице. Опција **ON BROWSER CLOSED** брише све податке из лоцал стораге-а само ако се бровсер затвори. Опција **CLEAR NOW** брише све податке из локалне меморије апликације приликом притиска на **APPLY CHANGES** или **EXIT AND SAVE CHANGES** у општим подешавањима.
- **APPLY CHANGES** – притиском на ову опцију све унете промене се аутоматски подешавају.
- **EXIT AND SAVE CHANGES** – притиском на ову опцију све унете промене се аутоматски подешавају и прозор општих подешавања се затвара.
- **EXIT**– ова опција затвара прозор општих подешавања апликације без снимања нових поставки за апликацију.

4.5. IV сегмент – канвас за исцртавање графика

Овај сегмент апликације налази се у доњем левом делу испод траке са алатима, и представља простор за графички приказ унетих функција. Максималне димензије канваса су прилагодјене уређају на којем се апликација користи за PC ова димензија износи ширина од 50px до 860px и висна од 50px до 520px. Димензије канваса могуће је подесити само на уводном екрану апликације. Као што је већ напоменуто боју канваса могуће је променити у општим подешавањима апликације или на почетку у уводном екрану. На слици 33 приказан је апликација са три графика исцртана на канвасу.



Слика 33. Пример употребе апликације за исцртавање више различитих графика

5. Закључак

Кроз овај рад описан је развој апликације **„FUNCTION GRAPH DRAWER“.** Ово је релативно једноставна апликација за исцртавање графика математичких функција. Развој апликације је базиран на webтехнологији, тачније на скриптном програмском језику JavaScript.

У уводном делу рада описане су основне технологије које су коришћене и дати су неки основни подаци везани за сам развој апликације. У другом поглављу рада представљени су UML дијаграми који су коришћени као алгоритамска шема при развоју апликације. Кроз њих се може видети како апликација комуницира са корисником и у каквој су међусобној релацији класе и методе апликације. У трећем поглављу дат је опис класа и функција унутар саме апликације, дат је увид у саму структуру појединих метода као и опис њиховог рада. Четврто поглавље даје детаљан опис корисничког интерфејса апликације као и упуство за њено коришћење.

Поред своје главне намене, што је очигледно исцртавање графика математичких функција, ова апликација може наћи примену при провери решења појединих функција. Апликација има доста простора за унапређење, постоји могућност одређивања тачки пресека између два или више графика, одређивање нула функције, одређивање домена функције итд.

Литература

Коришћено при изради писаног рада

[1] Jay Sridhar; "What Is JavaScript and How Does It Work? " интернет:https://www.makeuseof.com/tag/what-is-javascript/

[2] "Guide for using SASS preprocesing"; интернет: https://sass-lang.com/guide

[3] Филиповић Н.: *Алгоритми и структуре података,* Универзитет у Крагујевцу, Факултет инжењерских наука, Крагујевац, 2010.

Коришћено при изради апликације

[4] Библиотека векторске графике; интернет: https://fontawesome.com/

[5] Библиотека за евалуацију математичких израза; интернет: http://mathjs.org/

[6] Бесплатни фонт „Reckoner“ „Montserrat“; https://fonts.google.com/