KANDULA SRINIVASA REDDY MEMORIAL COLLEGE OF ENGINEERING
(AUTONOMOUS)

KADAPA-516003. AP

(Approved by AICTE, Affiliated to JNTUA, Ananthapuramu, Accredited by NAAC)

(An ISO 9001-2008 Certified Institution)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

VALUE ADDED COURSE

ON

"MIT APP INVENTOR"

Resource Person : Mr. J. Sunil, Assistant Professor, Dept. of AIML, KSRMCE

Course Coordinator: Mr. J. Sunil, Assistant Professor, Dept. of AIML, KSRMCE

Duration: 08/05/2023 to 20/05/2023

Lr./KSRMCE/AIML/2022-23 Date: 04-05-2023

To
The Principal,
KSRMCE,
Kadapa.

Respected Sir,

Sub: Permission to Conduct Value added Course on "MIT APP INVENTOR" **08/05/2023 to 20/05/2023**–Req - Reg.

The Department of Artificial Intelligence & Machine Learning is planning to offer a Value-Added Course on "MIT APP INVENTOR" to B. Tech. students. The course will be conducted from **08/05/2023 to 20/05/2023**. In this regard, I kindly request you to grant permission to conduct Value Added Course.

Thanking you sir,

Submitted to principal Sir,
KSRao

[illegible]
Yours faithfully

(Mr. J. Sunil, Asst. Professor in AIML)

Permitted
V. S. S. Murthy

K.S.R.M. COLLEGE OF ENGINEERING

(UGC-AUTONOMOUS)

Kadapa,Andhra Pradesh, India– 516 003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

An ISO 14001:2004 & 9001: 2015 Certified Institution

Cr./KSRMCE/AIML/2022-23/ Date: 05/05/2023

Circular

The Department of Artificial Intelligence & Machine Learning is offering a Value-Added Course on "MIT APP INVENTOR" from **08/05/2023 to 20/05/2023** to B. Tech students. In this regard, interested students are requested to register their names for the Value-Added Course for the below mentioned registration link.

https://forms.gle/utjX6PhCeWkD4Jse6

For further information contact Course Coordinator.

Course Coordinator: Mr. J. Sunil, Asst. professor, Dept. of AIML. -KSRMCE.
Contact No: 8978571543

[illegible]

HOD

Dept. of AIML

Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 005. (A.P.)

Cc to:

IQAC-KSRMCE

Registration for Value Added Course on MIT App Inventor from 08/05/2023 to 20/05/2023

* Required

1. Name

2. Roll Number *

3. Email *

4. Department *

Mark only one oval.

- ◯ CSE
- ◯ AIM
- ◯ L
  ECE

5. Semester *

Mark only one oval.

- ◯ I Sem
- ◯ II Sem
- ◯ III Sem
- ◯ IV Sem
- ◯ V Sem
- ◯ VI Sem
- ◯ VII Sem
- ◯ VIII Sem

Registration list for the Value Added Course on MIT App Inventor

Timestamp	Name	Roll Number	Department	Email	Semester
5-5-2023 13:27:26	A Amith	219y1a3904	AIML	219y1a3904@ksrmce.ac.in	IV
6-5-2023 23:15:49	A. Sharath Kumar Reddy	219y1a3902	AIML	219y1a3902@ksrmce.ac.in	IV
7-5-2023 11:22:33	Chenna venkatesh	219y1a3909	AIML	219y1a3909@ksrmce.ac.in	IV
5-5-2023 14:03:23	D .Vishnu Teja	219y1a3910	AIML	219y1a3910@ksrmce.ac.in	IV
7-5-2023 13:37:26	G. Nikhitha	219y1a3914	AIML	219y1a3914@ksrmce.ac.in	IV
8-5-2023 13:00:19	Janardhan Reddy Tamatam	219y1a05h3	CSE	219y1a05h3@ksrmce.ac.in	IV
6-5-2023 14:30:39	K Dhanush	229y5a3903	AIML	229y5a3903@ksrmce.ac.in	IV
6-5-2023 14:31:33	K Karthikan	219y1a3922	AIML	219y1a3922@ksrmce.ac.in	IV
8-5-2023 16:40:08	M Haritha	219y1a3928	CSE	219y1a3928@ksrmce.ac.in	IV
6-5-2023 15:07:14	M. Ranjith kumar	219y1a3925	AIML	219y1a3925@ksrmce.ac.in	IV
6-5-2023 19:22:33	N Guru Aakarsh	219y1a3932	AIML	219y1a3932@ksrmce.ac.in	IV
6-5-2023 17:25:26	N. Meghana	219y1a3934	AIML	219y1a3934@ksrmce.ac.in	IV
7-5-2023 17:22:33	N.Dhanush	219y1a05h9	CSE	219y1a05j8@ksrmce.ac.in	IV
5-5-2023 13:34:51	Nalipi Sridhar Reddy	219Y1A3933	AIML	219Y1A3933@ksrmce.ac.in	IV
8-5-2023 17:01:02	Nathikonda venkata venu	219y1a05c1	CSE	219y1a05c1@ksrmce.ac.in	IV
5-5-2023 14:14:43	P Kalyan Chakravarthi	219y1a3941	AIML	219y1a3941@ksrmce.ac.in	IV
6-5-2023 23:22:26	P Likitha	219y1a3937	AIML	219y1a3937@ksrmce.ac.in	IV
6-5-2023 23:22:12	P Pochamma	219y1a3943	AIML	219y1a3943@ksrmce.ac.in	IV
6-5-2023 23:25:23	P. Pravallika	219y1a3939	AIML	219y1a3939@ksrmce.ac.in	IV
6-5-2023 15:32:15	P.Sowjanya	219y1a3938	AIML	219y1a3938@ksrmce.ac.in	IV
6-5-2023 16:28:40	Patil pradeep	219y1a3942	AIML	219y1a3942@ksrmce.ac.in	IV
7-5-2023 15:48:31	PenubadiLakshminarayanamma	219y1a05d9	CSE	219y1a05d9@ksrmce.ac.in	IV
7-5-2023 13:49:26	R. Jaswitha	219y1a05e3	CSE	219y1a05e3@ksrmce.ac.in	IV
7-5-2023 11:32:27	R.Chandini	219y1a05e5	CSE	219y1a05e5@ksrmce.ac.in	IV
5-5-2023 13:31:31	S Deepak reddy	219y1a3911	AIML	219y1a3911@ksrmce.ac.in	IV
5-5-2023 14:01:12	S Hemanth Kumar	219y1a3955	AIML	219y1a3955@ksrmce.ac.in	IV
7-5-2023 11:47:21	S.pavani	219y1a05g4	CSE	219y1a05g4@ksrmce.ac.in	IV
7-5-2023 12:02:06	S.V.Rupa	219y1a05e7	CSE	219y1a05e7@ksrmce.ac.in	IV
6-5-2023 15:08:23	Santhosh Reddy	219y1a3912	AIML	219y1a3912@ksrmce.ac.in	IV
5-5-2023 13:33:40	Shaik Fysal Ahamed	219y1a3947	AIML	219y1a3947@ksrmce.ac.in	IV
7-5-2023 20:51:09	Shaik Noor Mahammad Sameeha	219y1a05f9	CSE	219y1a05f9@ksrmce .ac.in	IV
7-5-2023 21:28:40	Shashikala	219y1a05g7	CSE	219y1a05g7@ksrmce.ac.in	IV
6-5-2023 16:30:02	Sreenath	219y1a3956	AIML	219y1a3956@ksrmce.ac.in	IV
7-5-2023 15:02:48	T.Keerthi Reddy	219y1a05h6	CSE	219y1a05h6@ksrmce.ac.in	IV
8-5-2023 13:14:17	T.Nithya lavanya	219y1a3958	AIML	219y1a3958@ksrmce.ac.in	IV
7-5-2023 21:28:40	T.Priya	219y1a05h5	CSE	219y1a05h5@ksrmce.ac.in	IV
7-5-2023 20:37:26	V Sreelakshmi	219y1a05i4	CSE	219y1a05i4@ksrmce.ac.in	IV

7-5-2023 16:20:09	V.Drakshayini	219y1a05h9	CSE	219y1a05h9@ksrmce.ac.in	IV
8-5-2023 16:38:47	V.guru jahnavi	219y1a3963	AIML	219y1a3963@ksrmce.ac.in	IV
7-5-2023 14:31:06	Vaddemani Manasa	219y1a05i0	CSE	219y1a05i0@ksrmce.ac.in	IV
6-5-2023 20:49:55	Y. Janvi	219y1a3966	AIML	219y1a3966@ksrmce.ac.in	IV
7-5-2023 17:31:19	Y. Sneha latha	219y1a05j2	CSE	219y1a05j2@ksrmce.ac.in	IV
6-5-2023 18:27:06	Y.chenna kesava	219y1a3965	AIML	219y1a3965@ksrmce.ac.in	IV
7-5-2023 15:42:20	Yallala Chandrika	219y1a05i9	CSE	219y1a05i9@ksrmce.ac.in	IV

[illegible]
Co-ordinator

K S Rao
HOD

Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 005. (A.P.)

Syllabus of Value Added Course

Course Name: MIT APP INVENTOR

Course Objectives:

1. To enable adults to build apps using App Inventor (in a variety of potentially interdisciplinary settings, but definitely including App Challenge teams who want to complete their app).
2. App (Software) Design Process: How to effectively design and develop a concept and from it, an app (or other software).
3. Programming with App Inventor: How to use App Inventor as a tool within the design process; How to apply concepts & skills that will eventually allow build out of an app idea.

Course Outcomes: On Successful completion of this course the students will be able to,

1. Students will gain good understanding of using platforms like ai2 app inventor, thinkable, apply builder and other similar platforms
2. Students will learn about designing and debugging the app
3. Students will learn how to use Google cloud console to create a Google translate API key
4. Students will learn how to use blackly programming for developing an android app

UNIT-I

Introduction to MIT App Inventor, Simple Echo App, Overview of sensors in Android, Talk to me App, Ball bounce App, Mobile browser App, Paint pot app using paint component, simple animation app using animation component.

UNIT-II

Overview of Interface Components, Engineering, Debugging an App, Animations App, GPS Location tracking App, Google Doodle App, SMS Texting app, Ball bounce app, Bill amount app using buttons and text boxes.

UNIT-III

Overview of Layout components, Camcorder App, Demo app on sensors, Talk to me app using accelerometer, sensor, SMS Texting App, Table arrangements, Simple Calculator app using Vertical arrangements and Horizontal arrangements

UNIT-IV

Over of Media components: Contact picker App, No Text while driving App, Paris Map Tour App, and Lady bug Chase App, Make and take Quiz App, video player app, sound recorder app using text to speech.

UNIT-V

Over view of Sensor components, and storage components, Cloud DB, Tiny DB, storing data using spread sheet, Data base concept, Registration form App, Login page App, Calculator App, Registration App

Text Books/Reference Books:

1. Become an App Inventor: The Official Guide from MIT App Inventor(Your Guide to Designing, Building, and Sharing Apps)authors: MIT App Inventor; Karen Lang; MIT Computer Science and Artificial Intelligence Lab, Selim Tezel

2. Android Apps with App Inventor 2: Easy App Development for Everyone 1st Edition, by Karl-Hermann Rollke

3. App Inventor 2: Create Your Own Android Apps 2nd Edition
4. by David Wolber (Author), Hal Abelson (Author), Ellen Spertus (Author), Liz Looney (Author)

SCHEDULE

Department of Artificial Intelligence & Machine Learning

Value Added Course
On
"MIT App Inventor" From 08/05/2023 to 20/05/2023

Date	Timing	Resource Person	Topic to be covered
08/05/2023	9 AM to 12 PM	Mr. J. Sunil	Introduction to MIT App Inventor, Simple Echo App
08/05/2023	1 PM to 4 PM	Mr. J. Sunil	Overview of sensors in Android, Talk to me App, Ball bounce App
09/05/2023	9 AM to 12 PM	Mr. J. Sunil	Mobile browser App, Paintpot app using paint component
09/05/2023	1 PM to 4 PM	Mr. J. Sunil	Simple animation app using animation component.
10/05/2023	4 PM to 6 PM	Mr. J. Sunil	Overview of Interface Components, Engineering, Debugging an App, Animations App
11/05/2023	4 PM to 6 PM	Mr. J. Sunil	GPS Location tracking App Google Doodle App, SMS Texting app
12/05/2023	4 PM to 6 PM	Mr. J. Sunil	Ball bounce app, Bill amount app using buttons and text boxes.
13/05/2023	4 PM to 6 PM	Mr. J. Sunil	Overview of Layout components, Camcorder App, Demo app on sensors
15/05/2023	4 PM to 6 PM	Mr. J. Sunil	Talktome app using accelerometer, sensor, SMS Texting App, Table arrangements
16/05/2023	4 PM to 6 PM	Mr. J. Sunil	Simple Calculator app using Vertical arrangements and Horizontal arrangements
17/05/2023	4 PM to 6 PM	Mr. J. Sunil	Over of Media components: Contact picker App, No Text while driving App

18/05/2023	4 PM to 6 PM	Mr. J. Sunil	Paris Map Tour App, and Lady bug Chase App, Make and take Quiz App
19/05/2023	4 PM to 6 PM	Mr. J. Sunil	Video player app, sound recorder app using text to speech.
20/05/2023	4 PM to 6 PM	Mr. J. Sunil	Over view of Sensor components, and storage components, Cloud DB

[illegible] **Resource Person**

[illegible] **Coordinator**

[illegible] **HoD**

Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 005. (A.P.)

K.S.R.M. COLLEGE OF ENGINEERING

(UGC - AUTONOMOUS)

Kadapa, Andhra Pradesh,India - 516003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

An ISO 14001:2004 & 9001: 2015 Certified Institution

Department of Artificial Intelligence & Machine Learning

Value Added Course on MIT App Inventor

Attendance Sheet

S. No	Roll Num	Name of the Student	08/05/2023	09/05/2023	10/05/2023	11/05/2023	12/05/2023	13/05/2023	15/05/2023	16/05/2023	17/05/2023	18/05/2023	19/05/2023	20/05/2023
1	219y1a3904	A Amith	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2	219y1a3902	A Sharath Kumar Reddy	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
3	219y1a3909	Chenna venkatesh	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
4	219y1a3910	D .Vishnu Teja	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
5	219y1a3914	G. Nikhitha	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
6	219y1a05h3	Janardhan Reddy Tamatam	Jana	Jana	Jana	Jana	Jana	Jana	Jana	Jana	Jana	Jana	Jana	Jana
7	229y5a3903	K Dhanush	K. dhanush	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
8	219y1a3922	K Karthikan	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
9	219y1a3928	M Haritha	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10	219y1a3925	M. Ranjith kumar	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
11	219y1a3932	N Guru Aakarsh	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
12	219y1a3934	N. Meghana	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
13	219y1a05h9	N.Dhanush	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
14	219Y1A3933	Nalipi Sridhar Reddy	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
15	219y1a05c1	Nathikonda venkata venu	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
16	219y1a3941	P Kalyan Chakravarthi	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

17	219y1a3937	P Likitha	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
18	219y1a3943	P Pochamma	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
19	219y1a3939	P Pravallika	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
20	219y1a3938	P.Sowjanya	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
21	219y1a3942	Patil pradeep	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
22	219y1a05d9	PenubadiLakshminarayanamma	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
23	219y1a05e3	R. Jaswitha	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
24	219y1a05e5	R.Chandini	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
25	219y1a3911	S Deepak reddy	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
26	219y1a3955	S Hemanth Kumar	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
27	219y1a05g4	S.Pavani	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
28	219y1a05e7	S.V.Rupa	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
29	219y1a3912	Santhosh Reddy	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
30	219y1a3947	Shaik Fysal Ahamed	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
31	219y1a05f9	Shaik Noor Mahammad Sameeha	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
32	219y1a05g7	Shashikala	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
33	219y1a3956	Sreenath	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
34	219y1a05h6	T.Keerthi Reddy	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
35	219y1a3958	T.Nithya lavanya	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
36	219y1a05h5	T.Priya	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
37	219y1a05i4	V Sreelakshmi	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
38	219y1a05h9	V.Drakshayini	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
39	219y1a3963	V.guru jahnavi	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
40	219y1a05i0	Vaddemani Manasa	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
41	219y1a3966	Y. Janvi	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
42	219y1a05j2	Y. Sneha latha	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
43	219y1a3965	Y.Chenna kesava	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
44	219y1a05i9	Yallala Chandrika	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

[illegible]
Coordinator

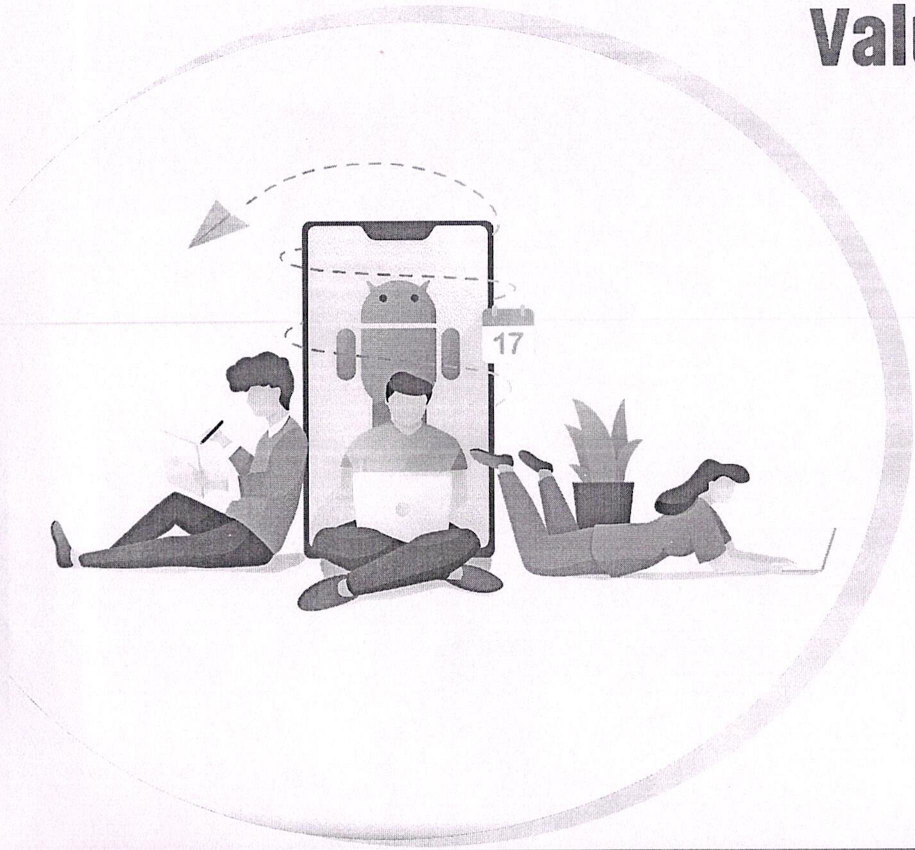
[illegible]
HOD

Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 005. (A.P.)

KSRM
COLLEGE OF ENGINEERING
(UGC - Autonomous)
Kadapa, Andhra Pradesh, India– 516 005
Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

KSNR
lives on..

Value Added course on MIT App Inventor



AI & ML

Python Programming Lab(AI-202)

08.05.2023 to 20.05.2023

Resource Person	**Mr. Sunil J** Asst. Prof, Department of AI & ML
Coordinator	**Mr. Sunil J** Asst. Prof, Department of AI & ML

Report of
Value Added Course on "MIT App Inventor"
From 08/05/2023 to 20/05/2023

Target Group	:	**B. Tech Students**
Details of Participants	:	**44 Students**
Co-coordinator(s)	:	**Mr. Sunil J, Assistant Professor, Dept of AI&ML**
Resource Person(s)	:	**Mr. Sunil J, Assistant Professor, Dept of AI&ML**
Organizing Department	:	**Artificial Intelligence & Machine Learning**
Venue	:	**Python Programming Lab (AI-202)**

Description:

The Department of Mechanical Engineering conducted a Value Added Course on "MIT App Inventor" from 08/05/2023 to 20/05/2023. The course Resource Persons is Mr. J Sunil, Assistant Professor, Dept. of AI&ML.

The main objective of this course is to introduce the fundamental concepts of MIT App Inventor is an intuitive, visual programming environment that allows everyone to build fully functional apps for Android phones, iPhones, and Android/iOS tablets. The MIT App Inventor can have a simple first app up and running in less than 30 minutes. And what's more, our blocks-based tool facilitates the creation of complex, high-impact apps in significantly less time than traditional programming environments. The MIT App Inventor project seeks to democratize software development by empowering all people, especially young people, to move from technology consumption to technology creation.

A small team of MIT CSAIL staff and students, led by Professor Hal Abelson, forms the nucleus of an international movement of inventors. In addition to leading educational outreach around MIT App Inventor and conducting research on its impacts, this core team maintains the free online app development environment that serves more than 6 million registered users.

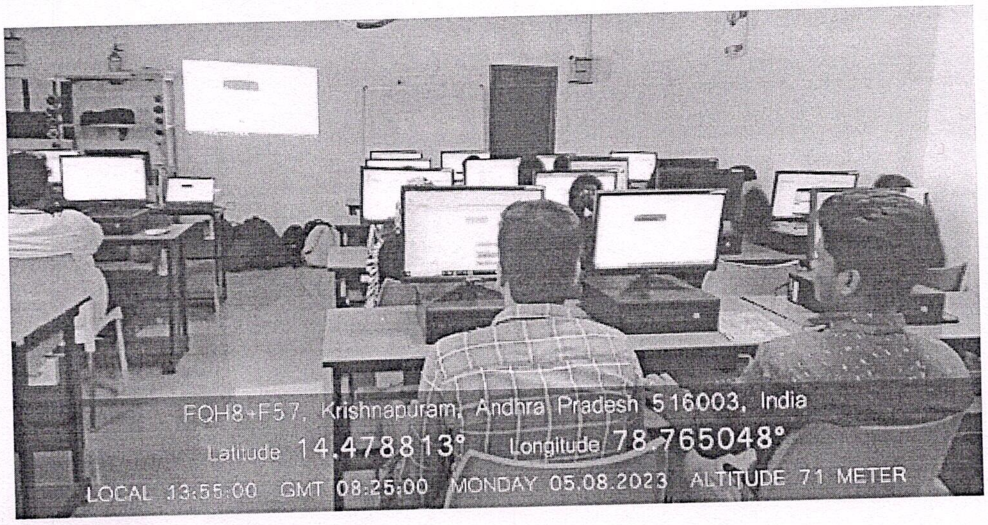
Blocks-based coding programs inspire intellectual and creative empowerment. MIT App Inventor goes beyond this to provide real empowerment for kids to make a difference – a way to achieve social impact of immeasurable value to their communities.

MIT App built in blocks:

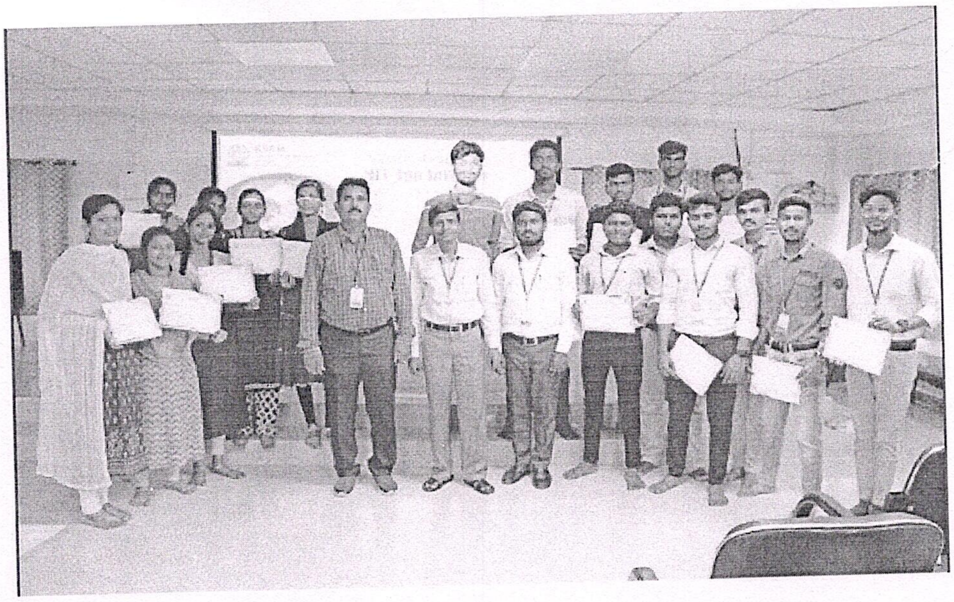
Built-in blocks are available regardless of which components are in your project. In addition to these language blocks, each component in your project has its own set of blocks specific to its own events, methods, and properties. This is an overview of all of the Built-In Blocks available in the Blocks Editor.

Photos

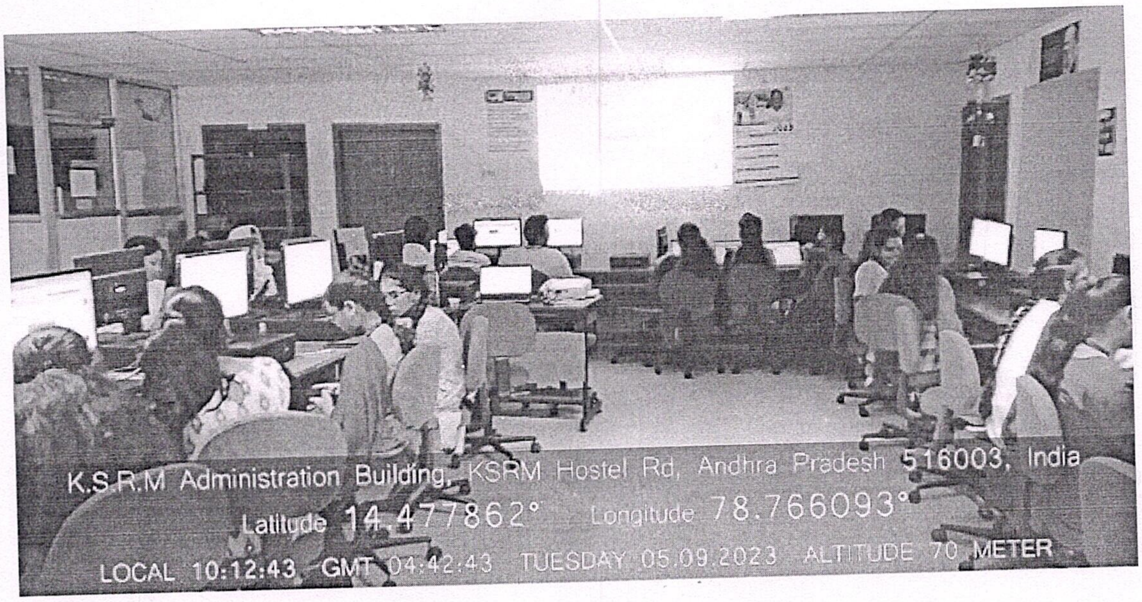
The pictures taken during the course are given below:


FQH8+F57, Krishnapuram, Andhra Pradesh 516003, India
Latitude 14.478813°
Longitude 78.765048°
LOCAL 13:55:00 GMT 08:25:00 MONDAY 05.08.2023 ALTITUDE 71 METER

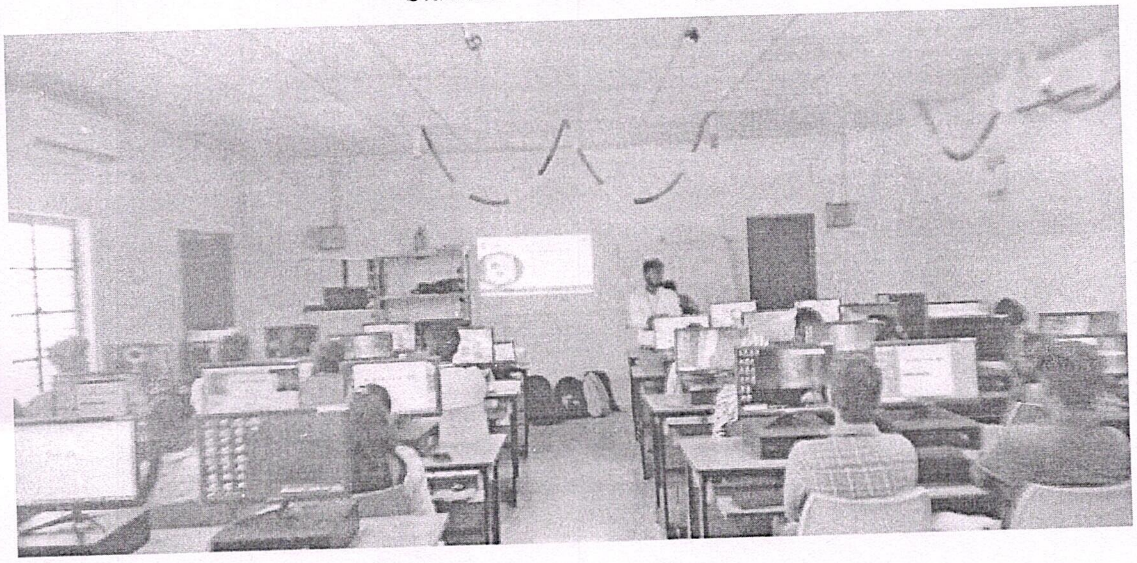
Students while practicing



Group picture with HoD's of AI&ML, CSE


K.S.R.M Administration Building, KSRM Hostel Rd, Andhra Pradesh 516003, India
Latitude 14.477862° Longitude 78.766093°
LOCAL 10:12:43 GMT 04:42:43 TUESDAY 05.09.2023 ALTITUDE 70 METER

Students while Practicing



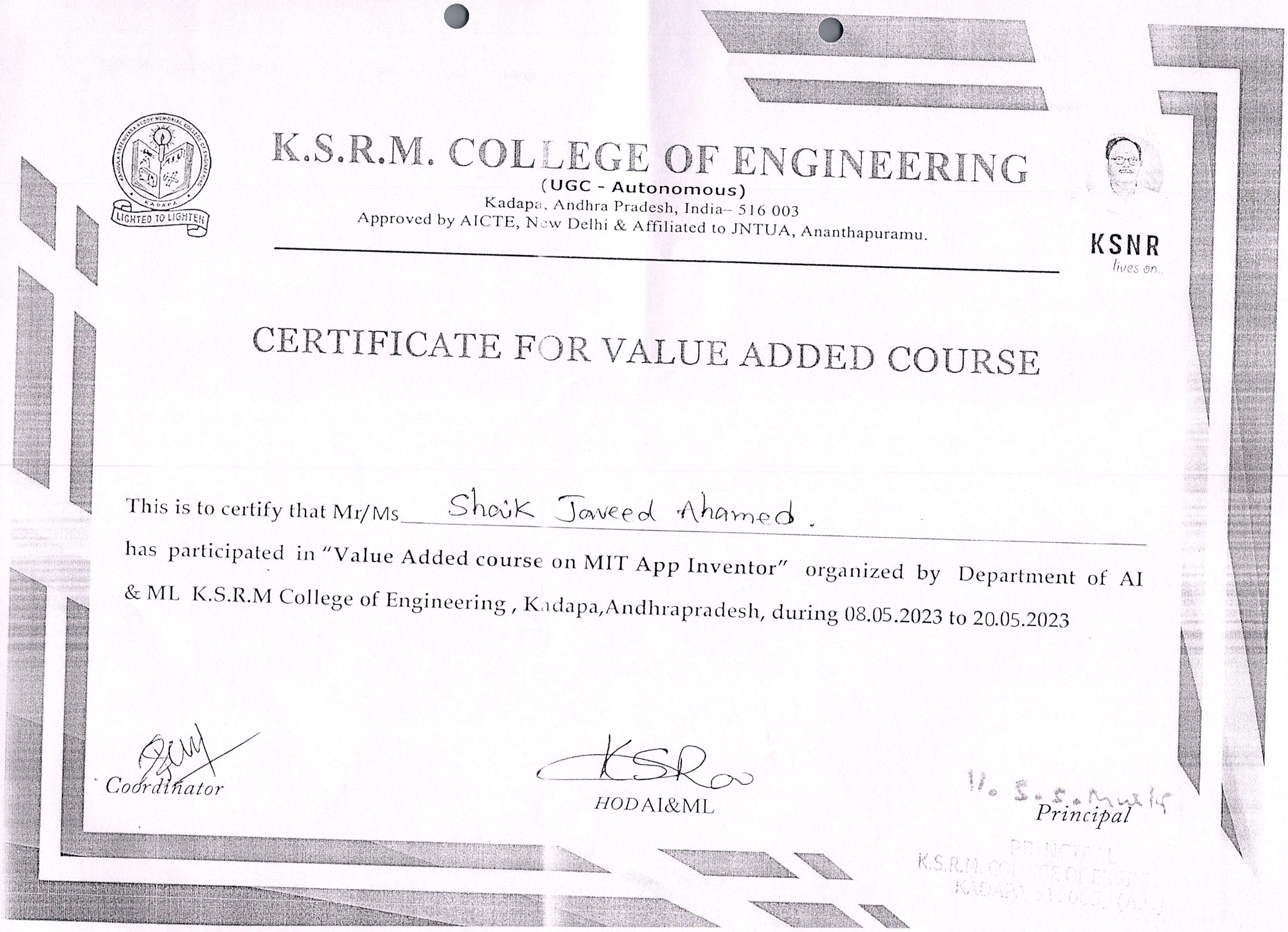
Lecture by Resource Person

[illegible]
Coordinator

[illegible]
HoD

Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 0[illegible]. (A.P.)

/ksrmce.ac.in Follow Us: /ksrmceofficial



K.S.R.M. COLLEGE OF ENGINEERING

(UGC - Autonomous)

Kadapa, Andhra Pradesh, India– 516 003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

LIGHTED TO LIGHTEN

KSNR lives on..

CERTIFICATE FOR VALUE ADDED COURSE

This is to certify that Mr/Ms Shaik Javeed Ahamed.

has participated in "Value Added course on MIT App Inventor" organized by Department of AI & ML K.S.R.M College of Engineering , Kadapa,Andhrapradesh, during 08.05.2023 to 20.05.2023

[illegible]

Coordinator

KSRao

HOD AI&ML

V. S. S. Murthy

Principal

PRINCIPAL
K.S.R.M. COLLEGE OF ENGINEERING
KADAPA-516 003. (A.P.)

K.S.R.M. COLLEGE OF ENGINEERING

(UGC - Autonomous)

Kadapa, Andhra Pradesh, India– 516 003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

LIGHTED TO LIGHTEN

KSNR

lives on..

CERTIFICATE FOR VALUE ADDED COURSE

This is to certify that Mr/Ms C. Sai Sree lakshmi

has participated in **"Value Added course on MIT App Inventor"** organized by Department of AI & ML K.S.R.M College of Engineering, Kadapa, Andhrapradesh, during 08.05.2023 to 20.05.2023

[illegible]

Coordinator

KSRao

HOD AI&ML

V. S. S. Murthy

Principal

PRINCIPAL
K.S.R.M. COLLEGE OF EN
KADAPA-516005

K.S.R.M. COLLEGE OF ENGINEERING

(UGC - Autonomous)

Kadapa, Andhra Pradesh, India– 516 003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

KSNR

lives on..

CERTIFICATE FOR VALUE ADDED COURSE

This is to certify that Mr/Ms P. B. V. Sravani

has participated in **"Value Added course on MIT App Inventor"** organized by Department of AI & ML K.S.R.M College of Engineering , Kadapa,Andhrapradesh, during 08.05.2023 to 20.05.2023

[illegible]

Coordinator

KSRao

HOD AI&ML

[illegible]

Principal

PRINCIPAL
K.S.R.M. COLLEGE OF ENGINEERING
KADAPA-516 003, (A.P.)

Feedback form on Value Added Course on "MIT App Inventor" from **08/05/2023 to 20/05/2023**

sunil.j@ksrmce.ac.in

* Required

Roll Number *

Your answer

Name of the Student *

Your answer

The objectives of the Value Added Course were met (Objective) *

- ○ Excellent
- ○ Good
- ○ Satisfactory
- ○ Poor

The content of the course was organized and easy to follow (Delivery) *

○ Excellent
○ Good
○ Satisfactory
○ Poor

The Resource Persons were well prepared and able to answer any question (Interaction) *

○ Excellent
○ Good
○ Satisfactory
○ Poor

The exercises/role play were helpful and relevant (Syllabus Coverage) *

○ Excellent
○ Good
○ Satisfactory
○ Poor

The Value Added Course satisfy my expectation as a value added Programme (Course Satisfaction) *

○ Excellent

○ Satisfactory

○ Good

○ Poor

Any Issues

Your answer

Submit

Clear form

K.S.R.M. COLLEGE OF ENGINEERING

Demartment of Artificial Intelligence & Machine Learning, Feedback report on Value Added Course on MIT App Inventor

Timestamp	Name	Roll Number	Email	Branch	The objectives of the Value Added Course were met	The content of the course was organized and easy to follow (Delivery) *	The Resource Persons were well prepared and able to	The exercises/ role play were helpful and	The Value Added Course satisfy my	Any Issu
5/20/2023 14:12:00	A Amith	219y1a3904	219y1a3904@ksrmce.ac.in	AIML	Excellent	Good	Satisfactory	Good	Good	Nc
5/20/2023 14:13:33	A. Sharath Kumar Reddy	219y1a3902	219y1a3902@ksrmce.ac.in	AIML	Good	Satisfactory	Good	Excellent	Excellent	
5/20/2023 14:16:23	Chenna venkatesh	219y1a3909	219y1a3909@ksrmce.ac.in	AIML	Excellent	Good	Good	Satisfactor	Good	
5/20/2023 14:16:30	D .Vishnu Teja	219y1a3910	219y1a3910@ksrmce.ac.in	AIML	Good	Good	Excellent	Good	Good	
5/20/2023 14:17:34	G. Nikhitha	219y1a3914	219y1a3914@ksrmce.ac.in	AIML	Satisfactory	Good	Excellent	Excellent	Good	
5/20/2023 14:17:35	Janardhan Reddy Tamatam	219y1a05h3	219y1a05h3@ksrmce.ac.in	AIML	Excellent	Good	Excellent	Good	Good	
5/20/2023 14:17:40	K Dhanush	229y5a3903	mce.ac.in	AIML	Excellent	Excellent	Good	Satisfactor	Good	
5/20/2023 14:18:20	K Karthikan	219y1a3922	219y1a3922@ksrmce.ac.in	AIML	Excellent	Satisfactory	Good	Excellent	Satisfactory	
5/21/2023 14:18:22	M Haritha	219y1a3928	219y1a3928@ksrmce.ac.in	AIML	Excellent	Satisfactory	Excellent	Excellent	Satisfactory	
5/21/2023 14:18:27	M. Ranjith kumar	219y1a3925	219y1a3925@ksrmce.ac.in	AIML	Satisfactory	Good	Satisfactory	Good	Satisfactory	
5/21/2023 15:19:40	N Guru Aakarsh	219y1a3932	219y1a3932@ksrmce.ac.in	AIML	Excellent	Good	Satisfactory	Good	Good	
5/21/2023 15:20:30	N. Meghana	219y1a3934	219y1a3934@ksrmce.ac.in	AIML	Good	Good	Good	Excellent	Good	
5/21/2023 15:41:42	N.Dhanush	219y1a05h9	219y1a05j8@ksrmce.ac.in	AIML	Good	Good	Satisfactory	Excellent	Excellent	
5/21/2023 16:20:22	Nalipi Sridhar Reddy	219Y1A3933	219Y1A3933@ksrmce.ac.in	AIML	Excellent	Good	Satisfactory	Good	Good	
5/21/2023 16:21:44	Nathikonda venkata venu	219y1a05c1	219y1a05c1@ksrmce.ac.in	AIML	Satisfactory	Satisfactory	Good	Good	Good	
5/21/2023 16:32:23	P Kalyan Chakravarthi	219y1a3941	219y1a3941@ksrmce.ac.in	AIML	Satisfactory	Excellent	Good	Excellent	Good	

5/22/2023 16:32:46	P Likitha	219y1a3937	219y1a3937@ksrmce.ac.in	AIML	Good	Good	Satisfactory	Satisfactor	Good
5/22/2023 16:40:20	P Pochamma	219y1a3943	219y1a3943@ksrmce.ac.in	AIML	Good	Satisfactory	Good	Good	Good
5/22/2023 16:44:32	P. Pravallika	219y1a3939	219y1a3939@ksrmce.ac.in	AIML	Good	Good	Excellent	Excellent	Excellent
5/22/2023 16:46:49	P.Sowjanya	219y1a3938	219y1a3938@ksrmce.ac.in	AIML	Good	Good	Satisfactory	Excellent	Excellent
5/22/2023 17:41:74	Patil pradeep	219y1a3942	219y1a3942@ksrmce.ac.in	AIML	Good	Good	Satisfactory	Excellent	Excellent
5/22/2023 17:43:51	PenubadiLakshminarayanamma	219y1a05d9	219y1a05d9@ksrmce.ac.in	CSE	Excellent	Good	Good	Good	Satisfactory
5/22/2023 17:41:52	R. Jaswitha	219y1a05e3	219y1a05e3@ksrmce.ac.in	CSE	Excellent	Good	Good	Excellent	Satisfactory
5/22/2023 17:43:35	R.Chandini	219y1a05e5	219y1a05e5@ksrmce.ac.in	CSE	Excellent	Good	Satisfactory	Excellent	Good
5/22/2023 17:51:54	S Deepak reddy	219y1a3911	219y1a3911@ksrmce.ac.in	CSE	Excellent	Satisfactory	Good		Good
5/22/2023 18:41:55	S Hemanth Kumar	219y1a3955	219y1a3955@ksrmce.ac.in	CSE	Good	Good	Good	Good	Good
5/22/2023 18:41:55	S.pavani	219y1a05g4	219y1a05g4@ksrmce.ac.in	CSE	Excellent	Excellent	Good	Good	Good
5/22/2023 18:41:55	S.V.Rupa	219y1a05e7	219y1a05e7@ksrmce.ac.in	CSE	Good	Excellent		Excellent	Excellent
5/22/2023 18:41:58	Santhosh Reddy	219y1a3912	219y1a3912@ksrmce.ac.in	CSE	Good	Satisfactory	Good	Good	Good
5/22/2023 18:44:59	Shaik Fysal Ahamed	219y1a3947	219y1a3947@ksrmce.ac.in	CSE	Good	Good	Good	Satisfactor	Good
5/23/2023 18:44:59	Shaik Noor Mahammad Sameeha	219y1a05f9	219y1a05f9@ksrmce .ac.in	CSE	Good	Good	Satisfactory	Good	Satisfactory
5/23/2023 18:45:30	Shashikala	219y1a05g7	219y1a05g7@ksrmce.ac.in	CSE	Excellent	Excellent	Excellent	Satisfactor	Excellent
5/23/2023 18:45:42	Sreenath	219y1a3956	219y1a3956@ksrmce.ac.in	CSE	Good	Excellent	Excellent	Good	Good
5/23/2023 18:46:20	T.Keerthi Reddy	219y1a05h6	219y1a05h6@ksrmce.ac.in	CSE	Good	Good	Good	Satisfactor	Good
5/23/2023 18:46:21	T.Nithya lavanya	219y1a3958	219y1a3958@ksrmce.ac.in	CSE	Good	Good	Excellent	Good	Excellent

5/23/2023 18:46:21	T.Priya	219y1a05h5	219y1a05h5@ksrmce.ac.in	CSE	Good	Good	Good	Good	Excellent	
5/23/2023 18:47:34	V Sreelakshmi	219y1a05i4	219y1a05i4@ksrmce.ac.in	CSE	Excellent	Good	Good	Satisfactory	Excellent	
5/23/2023 18:47:37	V.Drakshayini	219y1a05h9	219y1a05h9@ksrmce.ac.in	CSE	Good	Good	Excellent	Good	Good	
5/23/2023 18:47:42	V.guru jahnavi	219y1a3963	219y1a3963@ksrmce.ac.in	CSE	Good	Good	Excellent	Satisfactory	Excellent	
5/23/2023 18:47:50	Vaddemani Manasa	219y1a05i0	219y1a05i0@ksrmce.ac.in	CSE	Excellent	Satisfactory	Excellent	Satisfactory	Excellent	
5/23/2023 19:32:50	Y. Janvi	219y1a3966	219y1a3966@ksrmce.ac.in	CSE	Good	Satisfactory	Excellent	Satisfactory	Good	
5/23/2023 19:40:21	Y. Sneha latha	219y1a05j2	219y1a05j2@ksrmce.ac.in	CSE	Excellent	Excellent	Good	Good	Excellent	
5/23/2023 19:50:50	Y.chenna kesava	219y1a3965	219y1a3965@ksrmce.ac.in	CSE	Good	Good	Good	Excellent	Good	
5/23/2023 19:54:21	Yallala Chandrika	219y1a05i9	219y1a05i9@ksrmce.ac.in	CSE	Good	Good	Good	Good	Good	

[illegible]
Co-ordinator

KSRao
HOD

Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 005. (A.P.)

K.S.R.M. COLLEGE OF ENGINEERING (AUTONOMOUS), KADAPA-516003
DEPARTMENT OF ARTIFICIAL INTELIGENCE & MACHINE LEARNING
VALUE ADDED COURSE ON
MIT App Inventor FROM 08/05/2022 TO 20/05/2022
AWARD LIST

S.no	Name	Roll Number	Marks obtained
1	A.Nagaveni	229Y1A3901	18
2	A. Siva pradeep kumar	229Y1A3902	12
3	A.Md usman	229y1A3903	15
4	Barremukala Eswar	229y1a3904	12
5	B.Poojitha	229y1A3905	18
6	C Sai Sree Lakshmi	229y1A3906	20
7	Chadive Manasa	229Y1A3907	12
8	C. Indraja	229Y1A3908	16
9	Chappidi Avinash reddy	229Y1A3909	11
10	C Hema Priyanka	229Y1A3910	12
11	Jayasimha	229Y1A3911	20
12	C.Vyshnavi	229Y1A3912	19
13	Devisetty Bhavya Sree	229Y1A3913	19
14	G Chandrashekar	229y1a3915	09
15	Hari sree. M	229Y1A3916	7
16	K. Ganga bhavana	229y1a3918	18
17	Kanta Yamini	229y1a3919	19
18	K Sowmya	229Y1A3920	12
19	Kodathala Bhavana	229y1a3921	19
20	K.Abhilasha	229Y1A3922	20
21	Korapala Abhinaya sri	229Y1A3924	15
22	L Niranjan	229y1a3925	17
23	B.lakshmi vaishnavi	229Y1A3926	17
24	M.Umadevi	229y1a3927	18
25	M.Vijay Sai	229Y1A3928	19
26	M.Hitesh	229Y1A3929	20
27	Meruva Yamini	229y1a3931	20
28	Mughal Mohammed Ghani Baig	229Y1A3932	20
29	P. B. V. Sravani	229Y1A3936	19
30	P Girish	229y1a3938	14
31	Porumamilla kavya sree	229Y1A3939	12
32	P. Tulasi	229Y1A3940	10
33	P.Yasaswini	229Y1A3941	11
34	R Kasiviswanath Reddy	229Y1A3942	15
35	R Pavan Kumar	229y1a3943	19
36	R Sathvika	229Y1A3944	20
37	S Yasaswini	229Y1A3945	20
38	S Rajesh	229Y1A3946	20
39	Shaik Ayaan ahamed	229y1a3947	19
40	S Imam Khasim	229Y1A3948	19
41	Shaik Javeed Ahmed	229Y1A3950	20
42	Shaik Mohammed Zaheer	229y1a3953	18
43	Shaik Safiya Mehak	229Y1A3954	17
44	Shaik Shaheena Sultana	229Y1A3955	19

[illegible]
Coordinator

[illegible]
HoD

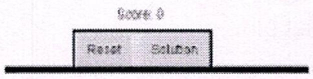
Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 005. (A.P.)

K.S.R.M. COLLEGE OF ENGINEERING (AUTONOMOUS), KADAPA-516003
DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING
VALUE ADDED COURSE ON
MIT APP INVENTOR FROM 08/05/2023 TO 20/05/2023
ASSESSMENT TEST

Roll Number: ____________________ **Name of the Student:** ________________________________

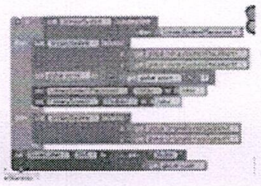
Time: 20 Min **(Objective Questions)** **Max. Marks: 20**

Note: Answer the following Questions and each question carries **one** mark.

1. 

What kind of arrangement is this

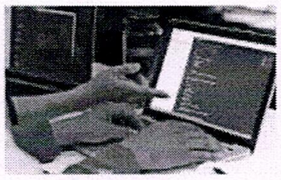
A Horizontal arrangement
B vertical arrangement
C diagonal arrangement

2. 
What is wrong with this block of code

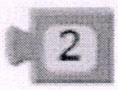
A It is missing the event handler block
B it is not going back to the original position
C It is not giving a point if it is put in the correct spot

3.
The screen where you can drag and drop component pieces and design them using the User Interface.

A Designer
B Blocks Editor
C H&M on 34th street next to the chicken and rice cart

4. 
The place where I tell the components of my app what to do

A Designer
B Blocks Editor
C Event
D Comment

5.  This is an example of what kind of data?

A String Data

B Number Data

C Boolean Data

D Words Data

6. By clicking the blue icon, the programmer can drag additional smaller blocks into the larger block, thus changing the shape and functionality of the original block.

A Mutator

B Global Variable

C Iteration

D Component

7. On the Palette, under which item will you find the Accelerometer?

A User Interface

B Layout

C Media

D Sensors

8. On the Palette, under which item will you find the Horizontal Arrangement?

A User Interface

B Layout

C Media

D Sensors

9. On the Palette, under which item will you find the Canvas?

A User Interface

B Layout

C Drawing and Animation

D Media

10. On the Blocks page, under which item will you find the "If-Then" blocks?

A Control

B Logic

C Math

D Text

11. The playing field for any App Inventor game is called a(n) _____?

A	Label	B	Canvas
C	Horizontal Alignment	D	Clock

12. All App Inventor Components have ______?

A	Labels	B	Images
C	Buttons	D	Properties

13. Where should a sprite be positioned when you first add it to your app?

A	anywhere you like	B	inside a canvas component
C	on the power cable	D	All of the above

14. Android is a _________

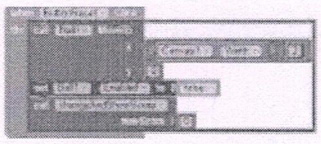
A	Open Source Software	B	Google Play Store
C	Flowchart	D	Sweet DISH

15. Non-visible component that can detect shaking and measure acceleration approximately in three dimensions using SI units

A	gyroscope sensor	B	pedometer
C	accelorometer	D	proximity

16. A formatting element in which to place components that should be displayed from left to right

A	Horizontal Arrangment	B	Table Arrangement
C	Vertical Arrangement	D	Sideways Arrangements

17. 

What does the purple block represent?

A Parameter

B Event Handler

C Command Block

D String

18. Can detect clicks.

A label

B button

C textbox

D slider

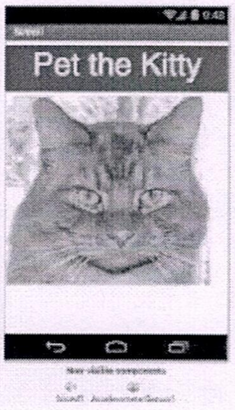
19. A component used to enter text

A Label

B Spinner

C Button

D Textbox

20. 

This section allows you to see the apps interface as it would look on your device.

A Media

B Animation

C Image Picker

D Viewer

Shaik Saheena sultana - 22941A395~~r~~

19

19/20

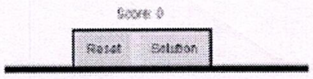
8/M

K.S.R.M. COLLEGE OF ENGINEERING (AUTONOMOUS), KADAPA-516003
DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING
VALUE ADDED COURSE ON
MIT APP INVENTOR FROM 08/05/2023 TO 20/05/2023
ASSESSMENT TEST

Roll Number: 22941A3958 **Name of the Student:** shaik saheena sultana.

Time: 20 Min **(Objective Questions)** **Max. Marks: 20**

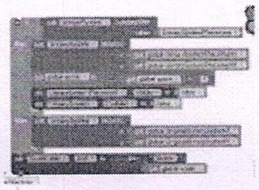
Note: Answer the following Questions and each question carries **one** mark.

1. 

What kind of arrangement is this ✓

A Horizontal arrangement [✓ B vertical arrangement]

C diagonal arrangement

2. 
What is wrong with this block of code ✓

A It is missing the event handler block [✓ B it is not going back to the original position]

C It is not giving a point if it is put in the correct spot

3.
The screen where you can drag and drop component pieces and design them using the User Interface. ✓

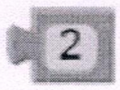
[✓ A Designer] B Blocks Editor

C H&M on 34th street next to the chicken and rice cart

4.
The place where I tell the components of my app what to do ✓

[✓ A Designer] B Blocks Editor

C Event D Comment

5. 

This is an example of what kind of data?

A String Data [✓ B] Number Data

C Boolean Data D Words Data

6. By clicking the blue icon, the programmer can drag additional smaller blocks into the larger block, thus changing the shape and functionality of the original block.

A Mutator [✓ B] Global Variable

C Iteration D Component

7. On the Palette, under which item will you find the Accelerometer?

[✓ A] User Interface B Layout

C Media D Sensors

8. On the Palette, under which item will you find the Horizontal Arrangement?

A User Interface B Layout

C Media [✓ D] Sensors

9. On the Palette, under which item will you find the Canvas?

A User Interface [✓ B] Layout

C Drawing and Animation D Media

10. On the Blocks page, under which item will you find the "If-Then" blocks?

A Control [✓ B] Logic

C Math D Text

11. The playing field for any App Inventor game is called a(n) _____?

[✓ A] Label

B Canvas

C Horizontal Alignment

D Clock

12. All App Inventor Components have ______?

A Labels

B Images

[✓ C] Buttons

D Properties

13. Where should a sprite be positioned when you first add it to your app?

A anywhere you like

[✓ B] inside a canvas component

C on the power cable

D All of the above

14. Android is a ________

A Open Source Software

B Google Play Store

C Flowchart

[✓ D] Sweet DISH

15. Non-visible component that can detect shaking and measure acceleration approximately in three dimensions using SI units

A gyroscope sensor

B pedometer

C accelorometer

[✓ D] proximity

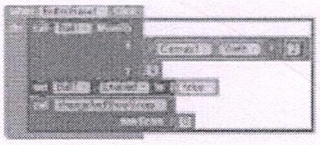
16. A formatting element in which to place components that should be displayed from left to right

A Horizontal Arrangment

[✓ B] Table Arrangement

C Vertical Arrangement

D Sideways Arrangements

17.  What does the purple block represent?

[✓ A] Parameter

B Event Handler

C Command Block

D String

18. Can detect clicks.

[✓ A] label

B button

C textbox

D slider

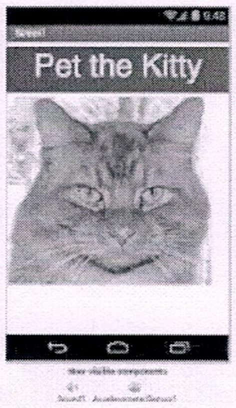
19. A component used to enter text

A Label

[✓ B] Spinner

C Button

D Textbox

20. 

This section allows you to see the apps interface as it would look on your device.

A Media

[✓ B] Animation

C Image Picker

D Viewer

Chappidi Avinash Reddy - 229Y1A3909

11

17/20

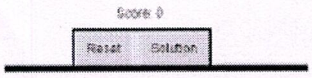
8sly

K.S.R.M. COLLEGE OF ENGINEERING (AUTONOMOUS), KADAPA-516003
DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING
VALUE ADDED COURSE ON
MIT APP INVENTOR FROM 08/05/2023 TO 20/05/2023
ASSESSMENT TEST

Roll Number: 229Y1A3909 **Name of the Student:** chappedi Avinash Reddy.

Time: 20 Min **(Objective Questions)** **Max. Marks: 20**

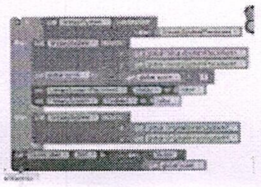
Note: Answer the following Questions and each question carries **one** mark.

1. 

What kind of arrangement is this

A Horizontal arrangement [✓ B vertical arrangement]

C diagonal arrangement

2. 
What is wrong with this block of code

A It is missing the event handler block [✓ B it is not going back to the original position]

C It is not giving a point if it is put in the correct spot

3.
The screen where you can drag and drop component pieces and design them using the User Interface.

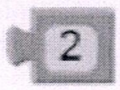
[✓ A Designer] B Blocks Editor

C H&M on 34th street next to the chicken and rice cart

4.
The place where I tell the components of my app what to do

A Designer [✓ B Blocks Editor]

C Event D Comment

5.  This is an example of what kind of data?

A String Data | [✓ B] Number Data
C Boolean Data | D Words Data

6. By clicking the blue icon, the programmer can drag additional smaller blocks into the larger block, thus changing the shape and functionality of the original block.

A Mutator | [✓ B] Global Variable
C Iteration | D Component

7. On the Palette, under which item will you find the Accelerometer?

A User Interface | B Layout
C Media | D Sensors

8. On the Palette, under which item will you find the Horizontal Arrangement?

A User Interface | B Layout
C Media | D Sensors

9. On the Palette, under which item will you find the Canvas?

A User Interface | B Layout
C Drawing and Animation | D Media

10. On the Blocks page, under which item will you find the "If-Then" blocks?

[✓ A] Control | B Logic
C Math | D Text

11. The playing field for any App Inventor game is called a(n) _____?

A Label B Canvas

C Horizontal Alignment D Clock

12. All App Inventor Components have _____?

A Labels B Images

[✓ C] Buttons D Properties

13. Where should a sprite be positioned when you first add it to your app?

A anywhere you like B inside a canvas component

C on the power cable D All of the above

14. Android is a ________

A Open Source Software B Google Play Store

C Flowchart [✓ D] Sweet DISH

15. Non-visible component that can detect shaking and measure acceleration approximately in three dimensions using SI units

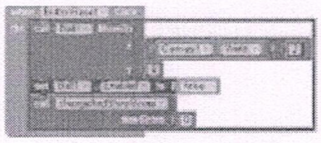
A gyroscope sensor B pedometer

C accelorometer D proximity

16. A formatting element in which to place components that should be displayed from left to right

A Horizontal Arrangment [✓ B] Table Arrangement

C Vertical Arrangement D Sideways Arrangements

17. 

What does the purple block represent?

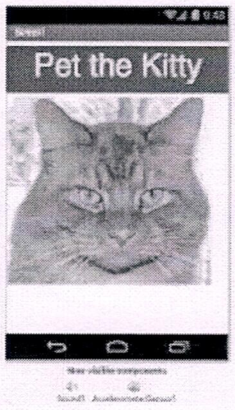
A	Parameter	B	Event Handler
C	Command Block	D	String

18. Can detect clicks.

[✓ A]	label	B	button
C	textbox	D	slider

19. A component used to enter text

A	Label	B	Spinner
C	Button	D	Textbox

20. 

This section allows you to see the apps interface as it would look on your device.

A	Media	[✓ B]	Animation
C	Image Picker	D	Viewer

K. Kasiviswanth Reddy - 229Y1A3942
15

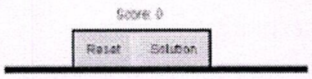
K.S.R.M. COLLEGE OF ENGINEERING (AUTONOMOUS), KADAPA-516003
DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING
VALUE ADDED COURSE ON
MIT APP INVENTOR FROM 08/05/2023 TO 20/05/2023
ASSESSMENT TEST

15/20 [illegible]

Roll Number: 229Y1A3942 **Name of the Student:** K. kaseveswanthreddy.

Time: 20 Min **(Objective Questions)** **Max. Marks: 20**

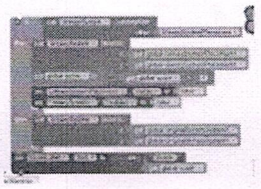
Note: Answer the following Questions and each question carries **one** mark.

1. 

What kind of arrangement is this ✗

[✓ A] Horizontal arrangement B vertical arrangement

C diagonal arrangement

2. 
What is wrong with this block of code ✗

A It is missing the event handler block B it is not going back to the original position

[✓ C] It is not giving a point if it is put in the correct spot

3.
The screen where you can drag and drop component pieces and design them using the User Interface. ✗

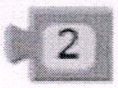
A Designer [✓ B] Blocks Editor

C H&M on 34th street next to the chicken and rice cart

4.
The place where I tell the components of my app what to do ✗

A Designer B Blocks Editor

[✓ C] Event D Comment

5.  This is an example of what kind of data?

A String Data

[✓ B] Number Data

C Boolean Data

D Words Data

6. By clicking the blue icon, the programmer can drag additional smaller blocks into the larger block, thus changing the shape and functionality of the original block.

A Mutator

B Global Variable

C Iteration

[✓ D] Component

7. On the Palette, under which item will you find the Accelerometer?

[✓ A] User Interface

B Layout

C Media

D Sensors

8. On the Palette, under which item will you find the Horizontal Arrangement?

A User Interface

B Layout

C Media

[✓ D] Sensors

9. On the Palette, under which item will you find the Canvas?

A User Interface

[✓ B] Layout

C Drawing and Animation

D Media

10. On the Blocks page, under which item will you find the "If-Then" blocks?

[✓ A] Control

B Logic

C Math

D Text

11. The playing field for any App Inventor game is called a(n) _____?

[✓ A] Label

B Canvas

C Horizontal Alignment

D Clock

12. All App Inventor Components have ______?

A Labels

B Images

[✓ C] Buttons

D Properties

13. Where should a sprite be positioned when you first add it to your app?

A anywhere you like

[✓ B] inside a canvas component

C on the power cable

D All of the above

14. Android is a ________

A Open Source Software

B Google Play Store

C Flowchart

[✓ D] Sweet DISH

15. Non-visible component that can detect shaking and measure acceleration approximately in three dimensions using SI units

A gyroscope sensor

B pedometer

C accelorometer

[✓ D] proximity

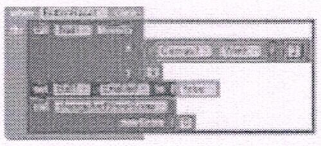
16. A formatting element in which to place components that should be displayed from left to right

A Horizontal Arrangment

[✓ B] Table Arrangement

C Vertical Arrangement

D Sideways Arrangements

17. 

What does the purple block represent? ✓

[✓ A] Parameter

B Event Handler

C Command Block

D String

18. Can detect clicks. ✓

[✓ A] label

B button

C textbox

D slider

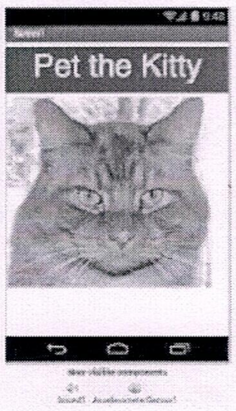
19. A component used to enter text ✓

A Label

[✓ B] Spinner

C Button

D Textbox

20. 

This section allows you to see the apps interface as it would look on your device. ✓

A Media

[✓ B] Animation

C Image Picker

D Viewer

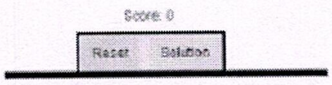
K.S.R.M. COLLEGE OF ENGINEERING (AUTONOMOUS), KADAPA-516003
DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING
VALUE ADDED COURSE ON
MIT APP INVENTOR FROM 08/05/2023 TO 20/05/2023
ASSESSMENT TEST

20/20

Roll Number: 22941A3911 **Name of the Student:** Jayasimha

Time: 20 Min **(Objective Questions)** **Max. Marks: 20**

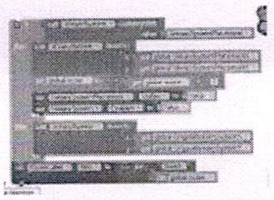
Note: Answer the following Questions and each question carries **one** mark.

1. 

What kind of arrangement is this

A Horizontal arrangement [✓ B] vertical arrangement

C diagonal arrangement

2. 
What is wrong with this block of code

A It is missing the event handler block [✓ B] it is not going back to the original position

C It is not giving a point if it is put in the correct spot

3.
The screen where you can drag and drop component pieces and design them using the User Interface.

[✓ A] Designer B Blocks Editor

C H&M on 34th street next to the chicken and rice cart

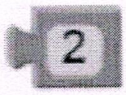
4. The place where I tell the components of my app what to do

[✓ A] Designer

B Blocks Editor

C Event

D Comment

5. 

This is an example of what kind of data?

A String Data

[✓ B] Number Data

C Boolean Data

D Words Data

6. By clicking the blue icon, the programmer can drag additional smaller blocks into the larger block, thus changing the shape and functionality of the original block.

A Mutator

[✓ B] Global Variable

C Iteration

D Component

7. On the Palette, under which item will you find the Accelerometer?

[✓ A] User Interface

B Layout

C Media

D Sensors

8. On the Palette, under which item will you find the Horizontal Arrangement?

A User Interface

B Layout

C Media

[✓ D] Sensors

9. On the Palette, under which item will you find the Canvas? ✓

A	User Interface	[✓ B]	Layout
C	Drawing and Animation	D	Media

10. On the Blocks page, under which item will you find the "If-Then" blocks? ✓

[✓ A]	Control	B	Logic
C	Math	D	Text

11. The playing field for any App Inventor game is called a(n) _____? ✓

[✓ A]	Label	B	Canvas
C	Horizontal Alignment	D	Clock

12. All App Inventor Components have _______? ✓

A	Labels	B	Images
[✓ C]	Buttons	D	Properties

13. Where should a sprite be positioned when you first add it to your app? ✓

A	anywhere you like	[✓ B]	inside a canvas component
C	on the power cable	D	All of the above

14. Android is a ________

A Open Source Software

B Google Play Store

C Flowchart

[✓ D] Sweet DISH

15. Non-visible component that can detect shaking and measure acceleration approximately in three dimensions using SI units

A gyroscope sensor

B pedometer

C accelorometer

[✓ D] proximity

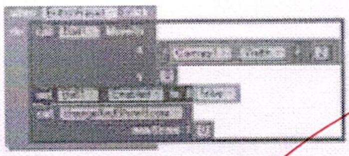
16. A formatting element in which to place components that should be displayed from left to right

A Horizontal Arrangment

[✓ B] Table Arrangement

C Vertical Arrangement

D Sideways Arrangements

17.  What does the purple block represent?

[✓ A] Parameter

B Event Handler

C Command Block

D String

18. Can detect clicks.

[✓ A] label

B button

C textbox

D slider

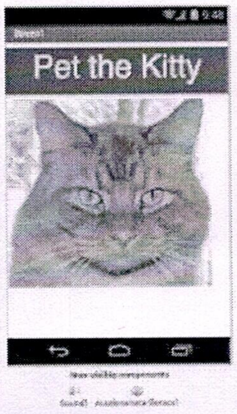
19. A component used to enter text

A Label

[✓ B] Spinner

C Button

D Textbox

20. 

This section allows you to see the apps interface as it would look on your device.

A Media

[✓ B] Animation

C Image Picker

D Viewer

M. Umadevi - 22941A3927

18

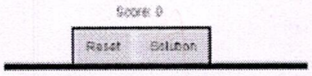
K.S.R.M. COLLEGE OF ENGINEERING (AUTONOMOUS), KADAPA-516003
DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING
VALUE ADDED COURSE ON
MIT APP INVENTOR FROM 08/05/2023 TO 20/05/2023
ASSESSMENT TEST

18/20 [illegible]

Roll Number: 22941A3927 **Name of the Student:** M · umadevi

Time: 20 Min **(Objective Questions)** **Max. Marks: 20**

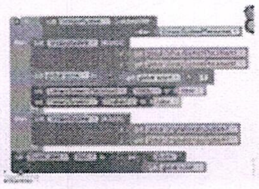
Note: Answer the following Questions and each question carries **one** mark.

1. 

What kind of arrangement is this ✓

A Horizontal arrangement [✓ B vertical arrangement]

C diagonal arrangement

2.  What is wrong with this block of code ✓

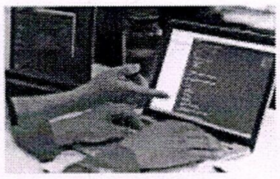
A It is missing the event handler block [✓ B it is not going back to the original position]

C It is not giving a point if it is put in the correct spot

3. The screen where you can drag and drop component pieces and design them using the User Interface. ✓

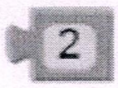
[✓ A Designer] B Blocks Editor

C H&M on 34th street next to the chicken and rice cart

4.  The place where I tell the components of my app what to do ✓

[✓ A Designer] B Blocks Editor

C Event D Comment

5. 

This is an example of what kind of data? ✓

A String Data

[✓ B] Number Data

C Boolean Data

D Words Data

6. By clicking the blue icon, the programmer can drag additional smaller blocks into the larger block, thus changing the shape and functionality of the original block. ✗

[✓ A] Mutator

B Global Variable

C Iteration

D Component

7. On the Palette, under which item will you find the Accelerometer? ✓

[✓ A] User Interface

B Layout

C Media

D Sensors

8. On the Palette, under which item will you find the Horizontal Arrangement? ✓

A User Interface

B Layout

C Media

[✓ D] Sensors

9. On the Palette, under which item will you find the Canvas? ✓

A User Interface

[✓ B] Layout

C Drawing and Animation

D Media

10. On the Blocks page, under which item will you find the "If-Then" blocks? ✓

[✓ A] Control

B Logic

C Math

D Text

11. The playing field for any App Inventor game is called a(n) _____?

[✓ A] Label B Canvas

C Horizontal Alignment D Clock

12. All App Inventor Components have _______?

A Labels B Images

[✓ C] Buttons D Properties

13. Where should a sprite be positioned when you first add it to your app?

A anywhere you like [✓ B] inside a canvas component

C on the power cable D All of the above

14. Android is a _________

A Open Source Software B Google Play Store

C Flowchart [✓ D] Sweet DISH

15. Non-visible component that can detect shaking and measure acceleration approximately in three dimensions using SI units

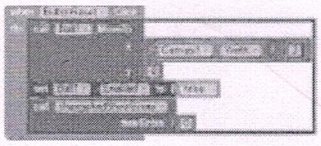
A gyroscope sensor B pedometer

C accelorometer [✓ D] proximity

16. A formatting element in which to place components that should be displayed from left to right

A Horizontal Arrangment [✓ B] Table Arrangement

C Vertical Arrangement D Sideways Arrangements

17. 

What does the purple block represent? ✓

[✓ A] Parameter

B Event Handler

C Command Block

D String

18. Can detect clicks. ✓

[✓ A] label

B button

C textbox

D slider

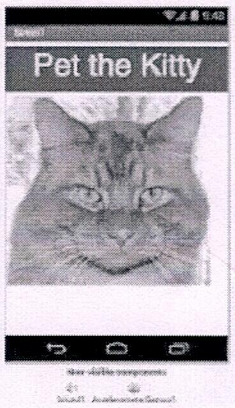
19. A component used to enter text ✗

A Label

B Spinner

[✓ C] Button

D Textbox

20. 

This section allows you to see the apps interface as it would look on your device. ✓

A Media

[✓ B] Animation

C Image Picker

D Viewer

MIT App Inventor: Objectives, Design, and Development

Evan W. Patton, Michael Tissenbaum and Farzeen Harunani

Abstract MIT App Inventor is an online platform designed to teach computational thinking concepts through development of mobile applications. Students create applications by dragging and dropping components into a design view and using a visual blocks language to program application behavior. In this chapter, we discuss (1) the history of the development of MIT App Inventor, (2) the project objectives of the project and how they shape the design of the system, and (3) the processes MIT uses to develop the platform and how they are informed by computational thinking literature. Key takeaways include use of components as abstractions, alignment of blocks with student mental models, and the benefits of fast, iterative design on learning.

Keywords Computational thinking · Computational action · Educational technology · Programming languages · Block-based programming · Mobile learning

3.1 Introduction

A smartphone is an information nexus in today's digital age, with access to a nearly infinite supply of content on the web, coupled with rich sensors and personal data. However, people have difficulty harnessing the full power of these ubiquitous devices for themselves and their communities. Most smartphone users consume technology without being able to produce it, even though local problems can often be solved with mobile devices. How then might they learn to leverage smartphone capabilities to solve real-world, everyday problems? MIT App Inventor is designed to democratize this technology and is used as a tool for learning computational thinking in a variety

E. W. Patton (✉) · M. Tissenbaum · F. Harunani
Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: ewpatton@csail.mit.edu; ewpatton@mit.edu

M. Tissenbaum
e-mail: miketissenbaum@gmail.com

F. Harunani
e-mail: farzeen@mit.edu

S.-C. Kong and H. Abelson (eds.), *Computational Thinking Education*,
https://doi.org/10.1007/978-981-13-6528-7_3

of educational contexts, teaching people to build apps to solve problems in their communities.

MIT App Inventor is an online development platform that anyone can leverage to solve real-world problems. It provides a web-based "What you see is what you get" (WYSIWYG) editor for building mobile phone applications targeting the Android and iOS operating systems. It uses a block-based programming language built on Google Blockly (Fraser, 2013) and inspired by languages such as StarLogo TNG (Begel & Klopfer, 2007) and Scratch (Resnick et al., 2009; Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010), empowering anyone to build a mobile phone app to meet a need. To date, 6.8 million people in over 190 countries have used App Inventor to build over 24 million apps. We offer the interface in more than a dozen languages. People around the world use App Inventor to provide mobile solutions to real problems in their families, communities, and the world. The platform has also been adapted to serve requirements of more specific populations, such as building apps for emergency/first responders (Jain et al., 2015) and robotics (Papadakis & Orfanakis, 2016).

In this chapter, we describe the goals of MIT App Inventor and how they have influenced our design and development—from the program's inception at Google in 2008, through the migration to MIT, to the present day. We discuss the pedagogical value of MIT App Inventor and its use as a tool to teach and encourage people of all ages to think and act computationally. We also describe three applications developed by students in different parts of the world to solve real issues in their communities. We conclude by discussing the limitations and benefits of tools such as App Inventor and proposing new directions for research.

3.2 MIT App Inventor Overview

[illegible] MIT App Inventor user interface includes two main editors: the design editor and the blocks editor. The design editor, or designer (see Fig. 3.1), is a drag and drop interface to lay out the elements of the application's user interface (UI). The blocks editor (see Fig. 3.2) is an environment in which app inventors can visually lay out the logic of their apps using color-coded blocks that snap together like puzzle pieces to describe the program. To aid in development and testing, App Inventor provides a mobile app called the App Inventor Companion (or just "the Companion") that developers can use to test and adjust the behavior of their apps in real time. In this way, anyone can quickly build a mobile app and immediately begin to iterate and test.

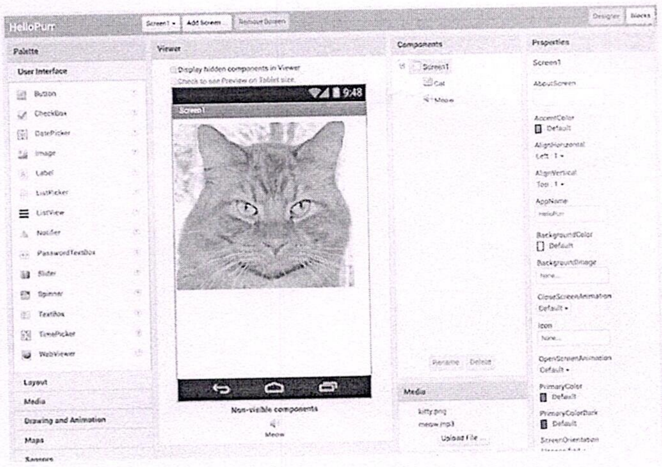


Fig. 3.1 App Inventor's design editor. App inventors drag components out from the palette (far left) to the viewer (center left) to add them to the app. Inventors can change the properties of the components (far right). An overview of the screen's components and project media are also displayed (center right)

3.3 MIT App Inventor Design Goals

In the design of MIT App Inventor, introducing mobile app development in educational contexts was a central goal. Prior to its release, most development environments for mobile applications were clunky, only accessible with expertise in systems level or embedded programming, or both. Even with Google's Android operating system and the Java programming language, designing the user interface was a complex task. Further, use of the platform required familiarity with Java syntax and semantics, and the ability to debug Java compilation errors (e.g., misspelled variables or misplaced semicolons) for success. These challenges presented barriers to entry for individuals not versed in computer science, App Inventor's target demographic. We briefly highlight and discuss design goals for the App Inventor project, specifically, the use of *components* to abstract some of the complexity of platform behavior, and the use of *blocks* to eliminate complexity of the underlying programming language. These goals can be further explained as aligning the visual language to the mental models of young developers and enabling exploration through fast, iterative design.

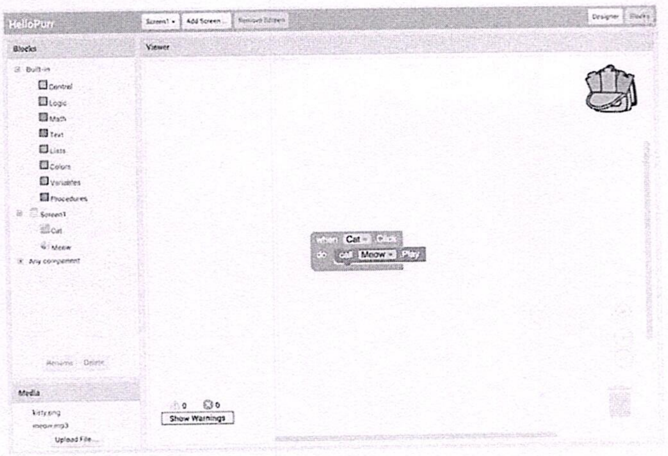


Fig. 3.2 App Inventor's blocks editor. Blocks code is typically read left to right, top to bottom. In this example, one would read "when Cat click, do call Meow play," that is, play the meow sound when the cat is clicked

3.3.1 Component Abstraction for Platform Behavior

Components are core abstractions in MIT App Inventor. Components reduce the complexity of managing interactions with platform-specific application programming interfaces (APIs) and details concerning state management of device hardware. This allows the user to think about the problem at hand rather than the minutia typically required of application developers. For example, someone planning to use MIT App Inventor to build an app to use the global positioning system (GPS) to track movement need not be concerned with application lifecycle management, GPS software and hardware locks, or network connectivity (in case location detection falls back to network-based location). Instead, the app developer adds a location sensor component that abstracts away this complexity and provides an API for enabling and processing location updates. More concretely, this implementation reduces 629 lines of Java code to 23 blocks, of which only two are required to accomplish location tracking. This reduction in complexity enables app inventors to focus on the problem at hand and quickly accomplish a goal.

Components are made up of three major elements: properties, methods, and events. *Properties* control the state of the component and are readable and/or writable by the app developer. For example, the enabled property of the location sensor includes the functionality required to configure the GPS receiver and to manage its state while the app is in use. *Methods* operate on multiple inputs and possibly return a result. *Events* respond to changes in the device or app state based on external factors. For example, when the app user changes their location, the location changed event allows the app logic to respond to the change.

3.3.2 Blocks as Logic

In MIT App Inventor, users code application behavior using a block-based programming language. There are two types of blocks in App Inventor: built-in blocks and component blocks. The built-in blocks library provides the basic atoms and operations generally available in other programming languages, such as Booleans, strings, numbers, lists, mathematical operators, comparison operators, and control flow operators. Developers use component blocks (properties, methods, and events) to respond to system and user events, interact with device hardware, and adjust the visual and behavioral aspects of components.

3.3.2.1 Top-Level Blocks

All program logic is built on three top-level block types: global variable definitions, procedure definitions, and component event handlers. Global variables provide named slots for storing program states. Procedures define common behaviors that can be called from multiple places in the code. When an event occurs on the device, it triggers the corresponding application behavior prescribed in the event block. The event handler block may reference global variables or procedures. By limiting the top-level block types, there are fewer entities to reason about.

3.3.3 Mental Modeling

The development team for App Inventor considered a number of restrictions when designing the environment. We examine a few design decisions, the rationale behind them, and their effects on computational thinking within App Inventor.

3.3.3.1 What You See Is What You Get (WYSIWYG)

The design editor for App Inventor allows developers to see how the app will appear on the device screen and adjust the form factor of the visualized device (e.g., phone or tablet). Adjustments to properties of the visual components, for example, background color and size, are reflected in real time. Apps can also be run in a *live development* mode using the Companion, which we will be discussed in more detail below.

The App Inventor team recently added capability for creating map-based applications. The functionality allows app inventors to drag, drop, and edit markers, lines, polygons, rectangles, and circles in their maps, as well as integrate web-based data from geographic information systems (GIS) to build content-rich apps. This way, the user can move the content around easily to achieve great results without needing to provide most of the logic for this in code.

3.3.3.2 Design Time Component Creation

Unlike many programming languages, App Inventor limits runtime creation of new entities. This provides multiple benefits. First, by explicitly positioning all components in the app, the user can visualize it clearly rather than having to reason about things that will not exist until a future time. Second, it reduces the chances of users introducing cyclic memory dependencies in the user interface that would eventually cause the app to run out of memory. This encourages app inventors to think about how to appropriately structure their applications and reuse components to avoid overloading the system or their end users.

3.3.3.3 Natural Numbering

The number system in App Inventor assumes a starting value of 1, in line with children's counting skills (Gelman & Gallistel, 1978). This is unlike most programming languages, which are more aligned with machine architecture and therefore start at 0.

4 *Fast Iteration and Design Using the Companion*

A key feature of MIT App Inventor is its live development environment for mobile applications. App Inventor provides this by means of a companion app installed on the user's mobile device. The App Inventor web interface sends code to the companion app, which interprets the code and displays the app in real time to the developer (Fig. 3.3). This way, the user can change the app's interface and behavior in real time. For example, a student making a game involving the ball component may want to bounce the ball off the edge of the play area. However, an initial implementation might have the ball collide with the wall and then stop. After discovering the Ball.EdgeReached event, the student can add the event and update the direction of the ball using the Ball.Bounce method. By testing the app and adjusting its programming in response to undesired behavior, students can explore more freely.

The traditional build cycle for an Android app involves writing code in a text editor or integrated development environment, and rebuilding the application for testing may often take minutes, whereas making a change in the live development

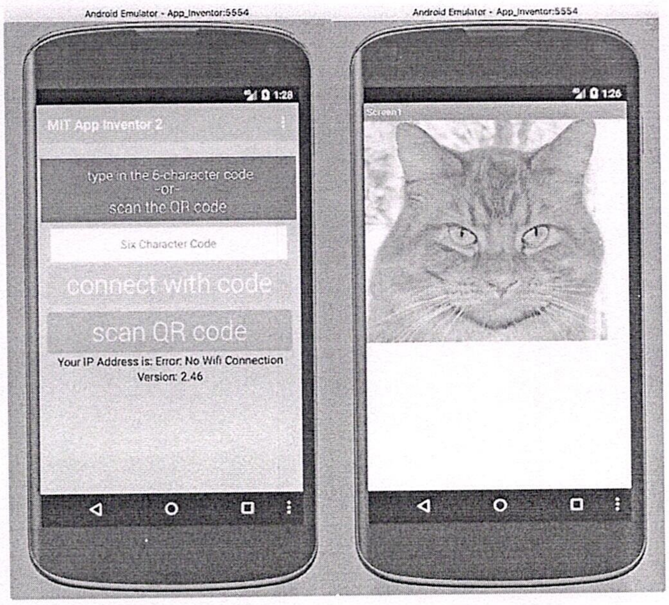

Android Emulator - App_Inventor:5554
MIT App Inventor 2
type in the 6-character code
-or-
scan the QR code
Six Character Code
connect with code
scan QR code
Your IP Address is: Error: No Wifi Connection
Version: 2.46
Android Emulator - App_Inventor:5554
Screen1

Fig. 3.3 The MIT Companion app interface for Android (left). After establishing a connection with the user's browser session, the active project is displayed in the companion app (right). See Fig. 3.1 for the designer view of the same project

environment typically takes effect in 1–2 s. Seeing changes reflected in the app quickly means that students can explore and even make mistakes while exploring, because the time cost of those mistakes is relatively small.

3.4 The History of MIT App Inventor

The App Inventor project began at Google in 2007 when Prof. Hal Abelson of MIT went on sabbatical at Google Labs. The project leads were inspired by increased interest in educational blocks programming languages, such as Scratch, and the release of the new Android operating system. This educational project was migrated to MIT when Google closed Google Labs in 2011. In this section, we briefly cover inception and early development of the App Inventor platform, first at Google, and then at MIT.

3.4.1 *Inception at Google*

Hal Abelson conceived the idea of App Inventor while on sabbatical at Google Labs in 2007. Abelson had previously taught a course at MIT on mobile programming, but at the time mobile app development required significant investment on the part of developers and development environments. Also in 2007, Google publicly announced the Android operating system. Abelson and Mark Friedman of Google began developing an intermediate language between the blocks language and Java APIs for Android, called Yet Another Intermediate Language (YAIL). The project was intended to help younger learners program for Android. Abelson and Friedman generated YAIL from a block-based language based on OpenBlocks (Roque, 2007), and the design of which was drawn from StarLogo TNG (Begel & Klopfer, 2007). The user interface and related components embodied Papert's idea of "powerful ideas in mind-size bites" (Papert, 1993). The Google version of the project terminated at the end of 2011, but the educational technology was transferred to MIT so that development and educational aspects could continue (Kincaid, 2011). Prof. Abelson joined Prof. Eric Klopfer of the Scheller Teacher Education Program lab and Prof. Mitch Resnick of the MIT Media Lab, forming a group called the MIT Center for Mobile Learning to carry on the App Inventor vision.

3.4.2 *Educational Expansion at MIT*

In late 2011, Google transferred stewardship of the App Inventor project to MIT. Much of the development focused on increasing capabilities to support educational goals of the project. At this time, the team developed additional curricula, making them freely available to teachers for computer science and computational thinking [illegible]ation. The MIT team also hosted a number of 1-day workshops, primarily around [illegible] northeast United States, training teachers in the pedagogy of App Inventor. We now focus on guided and open exploration in our materials rather than presenting students with step-by-step instructions in order to encourage self-guided learning. By making mistakes, students have the opportunity to practice more of the computational thinking principles, such as debugging, described by Brennan and Resnick (2012).

Technical development at MIT focused on development of new components including robotics (LEGO™ EV3), cloud-oriented data storage (CloudDB), and geographic visualization (Map). App Inventor team also developed Internet of Things related extensions so learners could interact with physical hardware external to their mobile devices, and to leverage the growing collection of small computer boards, such as Arduino, BBC micro:bit, and Raspberry Pi. To this day, the team continues its work of development, creating complementary educational materials in parallel.

3.5 MIT App Inventor in Education

The primary aim of MIT App Inventor is providing anyone with an interest in building apps to solve problems with the tools necessary to do so. Instructional materials developed by the team are primarily oriented toward teachers and students at the middle- and high-school levels, but app inventors come in all ages from around the world. In this section, we describe a few of the key components of the MIT App Inventor educational strategy, including massively online open courses (MOOCs) focused on MIT App Inventor, the Master Trainer (MT) program, the extensions functionality of App Inventor that allows incorporation of new material for education, and research projects that have leveraged App Inventor as a platform for enabling domain-specific computing.

3.5.1 Massive Open Online Courses

A desire to learn computational thinking has driven a proliferation of online educational material that anyone can access to increase their knowledge and understanding. As we continue to integrate information technology into our daily lives, mobile devices, and other new technologies, we can observe that a deeper understanding of computing is necessary to be an effective member of society, and those who learn computational thinking will have an advantage in our knowledge-driven economy.

Many massive open online courses have been developed wholely or in part using App Inventor. For example, an App Inventor EdX course closely integrates with the AP CS Principles course and incorporates many computational thinking elements. Students therefore can both build their own mobile apps and learn core competencies related to computation.

3.5.2 MIT Master Trainers Program

MIT provides special instruction to educators through the Master Trainers program.[1] A prototype of the Master Trainers program began during a collaboration with the Verizon App Challenge in 2012. Skilled App Inventor educators were recruited and given a small amount of special training to help mentor and train teams who subsequently won the App Challenge. The current Master Trainers program was conceived in 2015, to "grow a global community of experts on mobile app development who are available to guide others through the exploration of mobile app creation…, thus providing a pathway into computer science, software development, and other disciplines relevant in today's digital world."

[1] http://appinventor.mit.edu/explore/master-trainers.html.

In order to become a Master Trainer, one must demonstrate proficiency in App Inventor, for example, through taking the App Inventor EdX MOOC. The MOOC is highly integrated with computational thinking concepts, giving students a strong foundation in the concepts and practices associated with computational thinking. Aspiring Master Trainers then complete a 10-week online reading course covering topics such as App Inventor's mission and philosophy, pedagogy of teaching children and adults, constructionism, and design thinking. Lastly, there is an on-site 3-day workshop at MIT where participants dive into App Inventor features and learn to use App Inventor in a classroom to foster creativity, collaboration, and problem-solving. At the time of writing, there were 57 master trainers in 19 countries.

3.5.3 *Extensions*

Anyone with Java and Android programming experience can write their own components for App Inventor using our extension mechanism. For example, MIT recently published a suite of Internet of things (IOT)-related extensions[2] for interfacing with Arduino 101 and BBC micro:bit microcontrollers, with support for other platforms in development. Using these extensions, teachers can assemble custom curricula to leverage these technologies in the classroom and encourage their students to explore the interface between the world of software and the world of hardware.

We foresee the development of extensions related to artificial intelligence technologies, including deep learning, device support for image recognition, sentiment analysis, natural language processing, and more. Ideally, these complex technologies could be leveraged by anyone looking to solve a problem with the smartphone as a platform.

3.5.4 *Research Projects*

In addition to its pedagogical applications, App Inventor offers excellent opportunities for research in education and other areas. Early work focused on understanding how to appropriately name components for educational use (Turbak, Wolber, & Medlock-Walton, 2014). Usability in domain-specific contexts, such as humanitarian needs (Jain et al., 2015) and educational settings (Morelli, De Lanerolle, Lake, Limardo, Tamotsu, & Uche, 2011; Xie, Shabir, & Abelson, 2015), is also an area of interest. More recently, App Inventor has been used as a mechanism for data collection and visualization (Harunani, 2016; Mota, Ruiz-Rube, Dodero, & Figueiredo, 2016; Martin, Michalka, Zhu, & Boudelle, 2017). We are currently exploring expanding App Inventor's capabilities to include real-time collaboration between students, which should yield additional educational opportunities (Deng, 2017).

[2]http://iot.appinventor.mit.edu.

3.6 Empowerment Through Programming

By placing the output of student programming on mobile devices, App Inventor allows students to move their work out of traditional computer labs, and into their everyday lives and communities. This transition has powerful implications for what students create and how they envision themselves as digital creators. It allows students to shift their sense of themselves from individuals who "know how to code" to members of a community empowered to have a real impact in their lives and those of others. Below, we outline how App Inventor moves computing education from a focus on the theoretical to a focus on the practical, how we can reconceptualize computing education through a lens of computational action, and how we support students to engage in a broader community of digitally empowered creators.

3.6.1 From Theoretical to Practical

Traditional computer science curricula at the university level often focus on theory and include evaluation tools (e.g., Big-O notation of algorithms) and comprehension of the space and time complexity of data structures. Instead, App Inventor curricula focus on using a language practically to solve real-world problems. Rather than placing emphasis on learning concepts such as linked lists or key–value mappings, App Inventor hides the complexity of these data structures behind blocks so that students can spend more time designing apps that perform data collection and analysis, or integrate with a range of sensors and actuators interacting with external environments. This allows for a top-down, goal-based decomposition of the problem rather than a bottom-up approach, although App Inventor does not preclude such a strategy.

3.6.2 Computational Thinking

The concept of computational thinking was first used by Seymour Papert in his seminal book Mindstorms: Children, computers, and powerful ideas (1993); however, it was largely brought into the mainstream consciousness by Jeannette Wing in 2006. For Wing, computational thinking is the ability to think like a computer scientist. In the decade since, many educational researchers have worked to integrate computational thinking into modern computing and STEM curricula (Tissenbaum, Sheldon, & Sherman, 2018). However, the explosive growth of computational thinking has also resulted in a fragmentation of its meaning, with educational researchers, curriculum designers, and teachers using different definitions, educational approaches, and methods of assessments (Denning, 2017). There have been attempts to reconcile these differences (National Academy of Sciences, 2010) and to bring leading researchers together to compare and contrast these perspectives (Tissenbaum et al., 2018).

For most educational practitioners and researchers, computational thinking is dominated by an epistemological focus on computational thinking, in which students learn programming concepts (such as loops, variables, and data handling) and the use of abstractions to formally represent relationships between computing and objects in the real world (Aho, 2012). While this view has become the most prominent view of computational thinking, Papert critiqued mainstream schooling's emphasis on these "skills and facts" as a bias *against ideas* (Papert, 2000). Papert went further, arguing that students should be encouraged to follow their own projects and that learning the necessary skills and knowledge would arise as students encountered new problems and needed to solve (or not solve) them. This position of computational thinking and computing education fits more naturally with the ways that professionals engage in computer science: in pursuit of finishing a project, problems naturally come up and computer scientists reach out to the community through sites like Stack Overflow, or search the web for tutorials or other support. This disconnect between how we teach computing and how it is practiced in the real world requires us to critically reexamine theoretical and practical approaches. Below, we argue for an approach to computing education, termed computational action, that we believe matches these broader ideals.

3.6.3 Computational Action

While the growth of computational thinking has brought new awareness to the importance of computing education, it has also created new challenges. Many educational initiatives focus solely on the programming aspects, such as variables, loops, conditionals, parallelism, operators, and data handling (Wing, 2006), divorcing computing from real-world contexts and applications. This decontextualization threatens to make learners believe that they do not need to learn computing, as they cannot envision a future in which they will need to use it, just as many see math and physics education as unnecessary (Flegg et al., 2012; Williams et al., 2003).

This decontextualization of computing education from the actual lives of students is particularly problematic for students underrepresented in the fields of computing and engineering, such as women and other learners from nondominant groups. For these students, there is a need for their work to have an impact in their community and for it to help them develop a sense of fit and belonging (Pinkard et al., 2017). Lee and Soep (2016) argue that a critical perspective for computing is essential for students to develop a critical consciousness around what they are learning and making, moving beyond simply programming, instead of asking the students what they are programming and why they are programming it.

In response, the App Inventor team advocates for a new approach to computing education that we call *computational action*. The computational action perspective on computing argues that while learning about computing, young people should also have opportunities to create with computing which have direct impact on their lives and their communities. Through our work with App Inventor, we have developed two key dimensions for understanding and developing educational experiences that support students in engaging in computational action: (1) computational identity and (2) digital empowerment. Computational identity builds on prior research that showed the importance of young people's development of scientific identity for future STEM growth (Maltese & Tai, 2010). We define computational identity as a person's recognition that they can use computing to create change in their lives and potentially find a place in the larger community of computational problem-solvers. Digital empowerment involves instilling in them the belief that they can put their computational identity into action in authentic and meaningful ways.

Computational action shares characteristics with other approaches for refocusing computing education toward student-driven problem-solving, most notably *computational participation* (Kafai, 2016). Both computational action and computational participation recognize the importance of creating artifacts that can be used by others. However, there is a slight distinction between the conceptualizations of community in the two approaches. In computational participation, community largely means the broader community of learners engaging in similar computing practices (e.g., the community of Scratch programmers that share, reuse, and remix their apps). While such a learning community may be very beneficial to learners taking part in a computational action curriculum, the community of greater importance is the one that uses or is impacted by the learners' created products (e.g., their family, friends, and neighbors). This computational identity element of computational action acknowledges the importance of learners feeling a part of a computing community (i.e., those that build and solve problems with computing), but it is not a requirement that they actively engage with this larger community. A small group of young app builders, such as those described below, may develop significant applications and believe they are authentically part of the computing community, without having connected with or engaged with it in a deep or sustained way as would be expected in computational [illegible]icipation.

Through students' use of App Inventor, we have seen this computational action approach produce amazing results. Students in the United States have developed apps to help a blind classmate navigate their school (Hello Navi[3]); students in Moldova developed an app to help people in their country crowdsource clean drinking water (Apa Pura[4]); and as part of the CoolThink@JC project, students in Hong Kong created an app, "Elderly Guardian Alarm," to help the elderly when they got lost. Across these projects, we see students engaging with and facilitating change in their communities, while simultaneously developing computational identities.

[3]https://www.prnewswire.com/news-releases/321752171.html.

[4]The Apa Pura Technovation pitch video is available online at https://youtu.be/1cnLiSySizw.

3.6.4 *Supporting a Community Around Computation and App Creation*

We started the App of the Month program in 2015 in order to encourage App Inventors to share their work with the community. Any user can submit their app to be judged in one of four categories: Most Creative, Best Design, Most Innovative, and Inventor. Submissions must be App Inventor Gallery links, so that any user can remix winning apps. Furthermore, apps are judged in two divisions: youth and adult.

Now, 3 years after the program's inception, approximately 40 apps are submitted each month. More youth tend to submit than adults, and significantly more male users submit than female users, especially in the adult division. While submissions come in from all over the world, India and the USA are most highly represented.

Themes of submitted apps vary widely. Many students submit "all-in-one" apps utilizing the Text to Speech and Speech Recognizer components. Adults often submit learning apps for small children. Classic games, such as Pong, also get submitted quite frequently. Teachers tend to submit apps that they use in their classrooms.

Perhaps most importantly, students and adults alike submit apps designed to solve problems within their own lives or their communities. For example, a recent submitter noticed that the Greek bus system is subject to many slowdowns, so he built an app that tracks buses and their routes. Similarly, a student noticed that many of her peers were interested in reading books, but did not know how to find books they would like, so she built an app that categorizes and suggests popular books based on the Goodreads website.

However, not all users fit the same mold. One student found that he enjoys logic- and math-based games, and after submitting regularly for about a year, his skill improved tremendously. Hundreds of people have remixed his apps from the Gallery, and even downloaded them from the Google Play Store, encouraging the student to pursue a full-time career in game development.

The App of the Month program, as a whole, encourages users to think of App Inventor as a tool they can use in their daily lives and off-the-screen communities. It also provides incentive to share their apps and recognition for their hard work. Users go to App Inventor to solve problems—which makes them App Inventors themselves.

3.7 Discussion

We have seen in detail many aspects of the MIT App Inventor program from the development and educational perspective. There are some misconceptions, limitations, and benefits that are important to highlight.

3.7.1 Common Misconceptions

One common position detractors take is that blocks programming is not real programming (often comparing blocks languages to text languages). This is a false dichotomy if one understands programming to be the act of describing to a computer some realization of a Turing machine. The examples presented in earlier sections highlight how people use MIT App Inventor to solve real problems they face in their communities. To this end, younger individuals recognize that through tools such as App Inventor they can effect real change in their community, if not the whole world. Novice users who begin learning programming with blocks languages also tend to go further and continue more often than learners of textual languages (Weintrop & Wilensky, 2015).

Another common misconception is that creating mobile applications is something that only experts and those who have a lot of experience programming can do. However, students across the K-12 spectrum use App Inventor to develop their own mobile applications with little to no prior experience. For instance, the CoolThink@JC curriculum targets over 15,000 students in Hong Kong from grades 4–6. This intervention has enabled these elementary students to learn both to think computationally and to develop their own apps to address local issues (Kong et al., 2017).

3.7.2 Limitations

Computational proficiency is often assessed in traditional textual representations; for example, the AP Computer Science A exam is assessed in the Java programming language. For students who learn in block-based representations, it can be difficult to transition to textual representations. Therefore, it is important to help students transition to textual languages, while ensuring that knowledge gained in the visual language is not lost. Prof. Dave Wolber and a team at USF are actively addressing this through the development of the App Inventor Java Bridge.[5] The Java bridge allows anyone to translate an App Inventor application into a Java application compatible with Android Studio, the official text-based development environment used to build native Android applications. This enables students to transition from the AP Computer Science 0 curriculum to AP Computer Science A.

Another current limitation of App Inventor is that its design inhibits code reuse. Code reuse is one of the key computational thinking concepts in Brennan and Resnick's framework (2012). Many text-based languages provide robust support for code libraries and dependency management, allowing app developers to build on each other's work more easily. While App Inventor provides a gallery for publishing completed app source code, the community has yet to develop the granularity of libraries common in other programming languages. This presents an opportunity to

[5]http://appinventortojava.com/.

continue to grow the platform and user community and is a worthy subject for further exploration.

3.7.3 Benefits of Visual Programming for Mobile

Users of the App Inventor platform benefit from being able to repurpose the computational thinking skills they learn to interface with physical space in the external world. The visual programming of App Inventor and the abstraction and compartmentalization of concepts into components and blocks allow the app inventor to focus more on decomposing their problems into solvable elements. The facility of running apps on mobile devices allows the students to experience their own apps as part of an ecosystem they interact with daily, and with which they are intimately familiar. Since this encapsulation reduces the time it takes to build an app, even a straightforward prototype, app inventors can quickly grasp and iterate without paying a significant cost in terms of a compile-load-run cycle that is typical with mobile app development.

3.8 Conclusions

The MIT App Inventor project continues to push the boundaries of education within the context of mobile app development. Its abstraction of hardware capabilities and the reduction of complex logic into compact representations allows users to quickly and iteratively develop projects that address real-world problems. We discussed how App Inventor's curriculum development incorporates elements of computational thinking and encourages computational action with real-world effects. We also presented a number of projects that effectively accomplish this mission. We continue to grow the platform to democratize access to newer technologies, preparing future generations for a world in which computational thinking is a central part of problem-solving.

3.8.1 Future Vision

We already observe a rise in the growth of machine learning technologies. These technologies offer new ways of engaging with the world and could dramatically affect the future of technology and society. To support educating youth in this family of technologies, we are actively developing artificial intelligence and machine learning components, as well as curricula teachers can use to instruct students in these technologies.

In the future, we expect that households will be increasingly computationally literate. Already we are observing toddlers making use of devices such as phones and tablets to learn and engage the world in different ways. These technologies will become nearly universal in the near future, mandating increased pedagogy around computational thinking as well as the creation of environments to aid young children in using these tools to solve problems is critical. We must help them become producers and change makers rather than simply consumers. Increasingly, we move toward a world where functionality is abstracted, or provided as pieces that the computationally literate can combine for novel solutions for any problem. App Inventor will continue to push these boundaries by exploring bleeding edge technologies and integrating them within a mobile context.

Lastly, we are moving toward economies of knowledge. In these economies, access to new innovations and ways of solving problems will differentiate individuals competing globally (Powell & Snellman, 2004). Tools that provide increased abstraction for solving problems will offer more advantages to individuals than traditional engineering approaches.

Acknowledgements The authors would like to thank Prof. Hal Abelson, Karen Lang, and Josh Sheldon for their input, and discussions of material in the manuscript. App Inventor has received financial support from Google, NSF grant #1614548, Hong Kong Jockey Club, Verizon Foundation, and individual contributors.

References

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832–835.

Begel, A., & Klopfer, E. (2007). Starlogo TNG: An introduction to game development. *Journal of E-Learning, 53*, 146.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In: *Proceeding of the 2012 AERA*.

Deng, X. (2017). *Group collaboration with App Inventor* (Masters thesis, Massachusetts Institute of Technology).

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33–39.

Flegg, J., Mallet, D., & Lupton, M. (2012). Students' perceptions of the relevance of mathematics in engineering. *International Journal of Mathematical Education in Science and Technology, 43*(6), 717–732.

Fraser, N. (2013). Blockly: A visual programming editor. https://developers.google.com/blockly/.

Gelman, R., & Gallistel, C. R. (1978). *The child's understanding of number*. Cambridge, MA: Harvard University Press.

Harunani, F. (2016). AppVis: Enabling data-rich apps in App Inventor. Masters Thesis, University of Massachusetts, Lowell.

Jain, A., Adebayo, J., de Leon, E., Li, W., Kagal, L., Meier, P., et al. (2015). Mobile application development for crisis data. *Procedia Engineering, 107*, 255–262.

Kafai, Y. B. (2016). From computational thinking to computational participation in K-12 education. *Communications of the ACM, 59*(8), 26–27.

Kincaid, J. (2011). Google gives Android App Inventor a new home at MIT Media Lab. Techcrunch. Retrieved March 04, 2018, from https://techcrunch.com/2011/08/16/google-gives-android-app-inventor-a-new-home-at-mit-media-lab/.

Kong, S., Abelson, H., Sheldon, J., Lao, A., Tissenbaum, M., & Lai, M. (2017). Curriculum activities to foster primary school students' computational practices in block-based programming environments. In *Proceedings of Computational Thinking Education* (p. 84).

Lee, C. H., & Soep, E. (2016). None but ourselves can free our minds: critical computational literacy as a pedagogy of resistance. *Equity & Excellence in Education, 49*(4), 480–492.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE), 10*(4), 16.

Maltese, A. V., & Tai, R. H. (2010). Eyeballs in the fridge: Sources of early interest in science. *International Journal of Science Education, 32*(5), 669–685.

Martin, F., Michalka, S., Zhu, H., & Boudelle, J. (2017, March). Using AppVis to build data-rich apps with MIT App Inventor. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 740–740). ACM.

Morelli, R., De Lanerolle, T., Lake, P., Limardo, N., Tamotsu, E., & Uche, C. (2011, March). Can Android App Inventor bring computational thinking to k-12. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE'11)* (pp. 1–6).

Mota, J. M., Ruiz-Rube, I., Dodero, J. M., & Figueiredo, M. (2016). Visual environment for designing interactive learning scenarios with augmented reality. *International Association for Development of the Information Society*.

National Academies of Science. (2010). *Report of a workshop on the scope and nature of computational thinking*. Washington DC: National Academies Press.

Papadakis, S., & Orfanakis, V. (2016, November). The combined use of Lego Mindstorms NXT and App Inventor for teaching novice programmers. In *International Conference EduRobotics 2016* (pp. 193–204). Springer, Cham.

Papert, S. (1990). *A critique of technocentrism in thinking about the school of the future*. Cambridge, MA: Epistemology and Learning Group, MIT Media Laboratory.

Papert, S. (1993). *Mindstorms: Children, computers, and powerful ideas* (2nd ed.). Basic Books.

Papert, S. (2000). What's the big idea? Toward a pedagogy of idea power. *IBM Systems Journal, 39*(3–4), 720–729.

Pinkard, N., Erete, S., Martin, C. K., & McKinney de Royston, M. (2017). Digital Youth Divas: Exploring narrative-driven curriculum to spark middle school girls' interest in computational activities. *Journal of the Learning Sciences*, (just-accepted).

Powell, W. W., & Snellman, K. (2004). The knowledge economy. *Annual Reviews in Sociology, 30,* 199–220.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60–67.

Roque, R. V. (2007). OpenBlocks: *An extendable framework for graphical block programming systems*. Doctoral dissertation, Massachusetts Institute of Technology.

Tissenbaum, M., Sheldon, J., & Sherman, M. (2018). The state of the field in computational thinking assessment. In *To Appear in the Proceedings of the 2018 International Conference of the Learning Sciences*. London.

Turbak, F., Wolber, D., & Medlock-Walton, P. (2014, July). The design of naming features in App Inventor 2. In *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 129–132). IEEE.

Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: Students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children (IDC'15)* (pp. 199–208).

Williams, C., Stanisstreet, M., Spall, K., Boyes, E., & Dickson, D. (2003). Why aren't secondary students interested in physics? *Physics Education, 38*(4), 324.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Xie, B., Shabir, I., & Abelson, H. (2015, October). Measuring the usability and capability of app inventor to create mobile applications. In *Proceedings of the 3rd International Workshop on Programming for Mobile and Touch* (pp. 1–8). ACM.