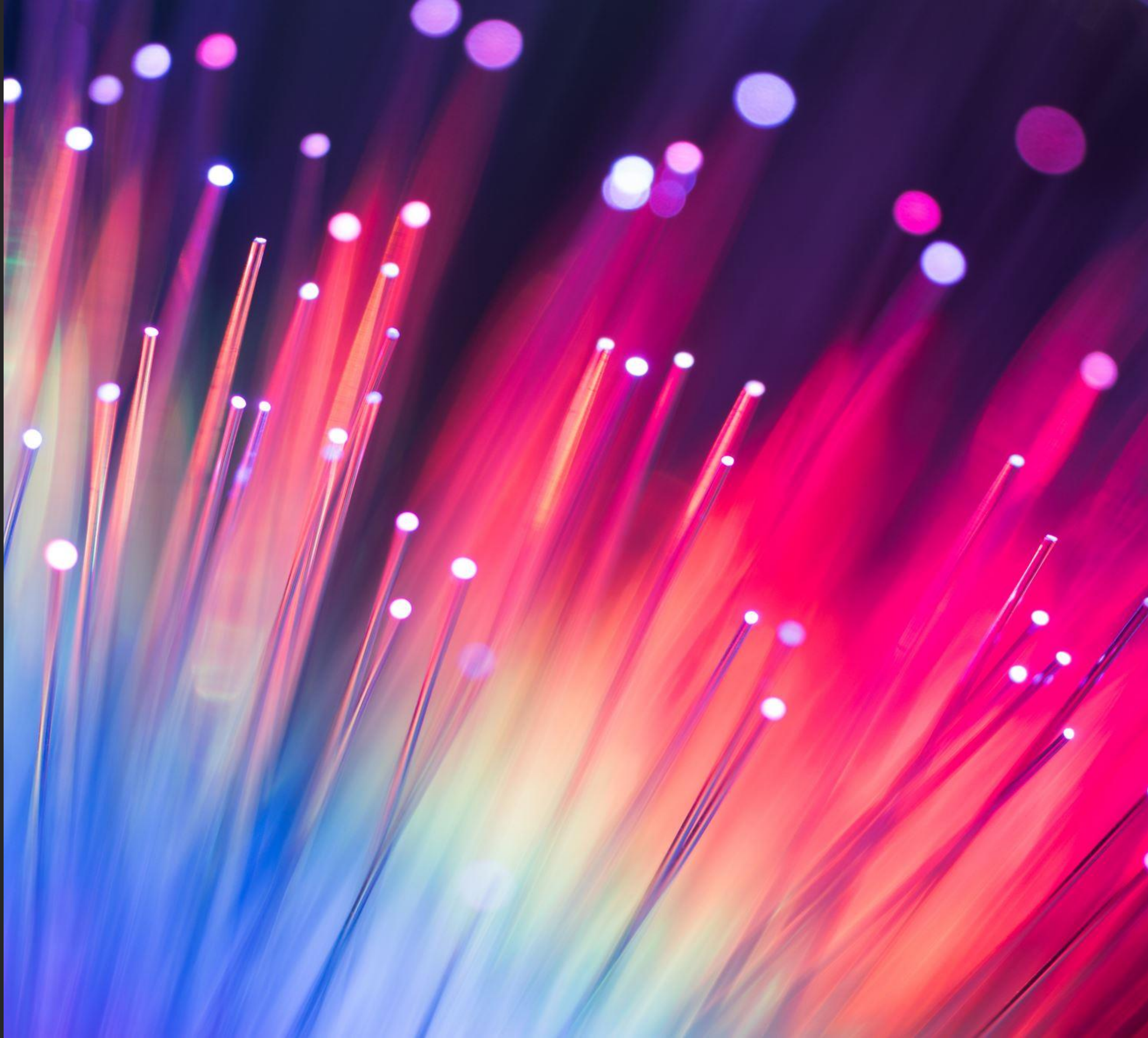
PHASE II FINAL

FINAL DEMO

TEAM RSSF

OCT 2019



SYSTEM ADVANTAGES

- The Blind Buddy Visually Impaired Navigational Assistance Application (BBVINAA) is robust, modular, and expandable.
- Simplistic and easy to use setup system.
- Accessible and made specifically with visually impaired individuals in mind.

ROBUST

- Since users will rely on the apps ability to continuously function as a navigational assistant it is key that it does not break or shutdown during use.
- Built using the Google Maps Platform which has been trusted and utilized by many.
- System uses error catching to ensure that the app continues to run and assist the user in the unlikely event that an error does occur.

MODULAR

- System divided into multiple components, which not only lowers the likelihood of errors, but also lessens the impact of errors if they occur.
- Modular design allows for much more flexibility of the application so changes can be made easily and efficiently.

EXPANDABLE

- Made with future updates and improvements to app in mind.
- Easy to integrate newer technologies, maps, and location providers.
- Continuous improvements will create a greatly improved user experience.

SCOPE CREEP (PREVIOUS)

- Because of the difficulty of the semester, our team has almost no room for scope creep.
- Future unseen challenges predicted in building this app may prove very difficult.
  - Team has little experience with mobile development.
  - Possibilities with third-party software such as Google Indoor Maps unknown.
  - General clearness that emerges with the development process will reveal unforeseen challenges.
- We will do our best to keep up with it.

SCOPE CREEP (CURRENT)

- Difficult semester left little room for scope creep as predicted
- Result of unseen challenges
  - Mobile development
  - Google Maps platform
  - Mapwize API
  - Emulator limitations

FUNCTION POINTS

FUNCTIONAL REQUIREMENTS

P FR_ ID	*Preliminary FR Description*
PFR1	Accepting from the user the destination location to go
PFR2	Figuring out the routes to reach each destination
PFR3	Informing the user of the routes to reach the destination
PFR4	Informing the user to walk a certain distance
PFR5	Informing the user to stop at the right place to turn
PFR6	Detecting obstacles and informing the user how to avoid them
PFR7	Placing emergency calls and messages
PFR8	Detecting when the user falls
PFR9	Predict the user’s next actions based on the user’s schedule and habits

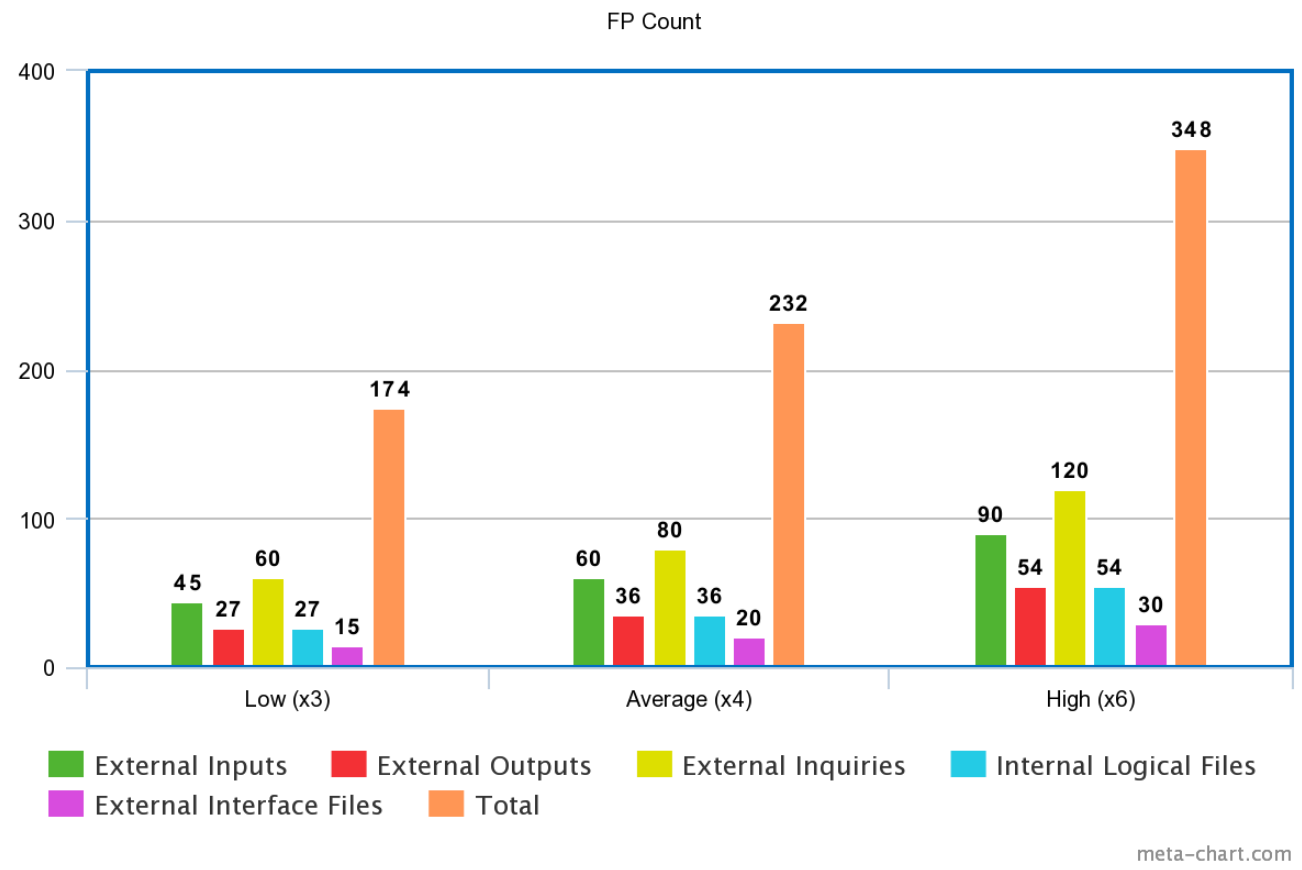
COMPLEXITY RATE

Functional Requirement	*Category*	*Complexity (Rate 0-10)*
PFR1	External Inputs	5
PFR2	Internal Logical Files	9
PFR3	External Outputs	3
PFR4	External Outputs	3
PFR5	External Outputs	3
PFR6	External Inquiries	10
PFR7	External Interface files	5
PFR8	External Inputs	10
PFR9	External Inquiries	10

FP COUNT

Type of Component	*Complexity of Components*		
	Low (x3)	Average (x4)	High (x6)
External Inputs	45	60	90
External Outputs	27	36	54
External Inquiries	60	80	120
Internal Logical Files	27	36	54
External Interface files	15	20	30
Total	174	232	348

FP COUNT


FP Count
400
300
200
100
0
45
27
60
27
15
174
60
36
80
36
20
232
90
54
120
54
30
348
Low (x3)
Average (x4)
High (x6)
🟩 External Inputs
🟥 External Outputs
🟨 External Inquiries
🟦 Internal Logical Files
🟪 External Interface Files
🟧 Total
meta-chart.com