

Quick Recap

Your meetings, summarized in seconds.

By : Team 2
Bros of Balayya

Agenda

1. Problem Statement
2. Project Description
3. Persona
4. MVP
5. Technologies & Algorithm
6. Project Schedule
7. Teamwork Agreement
8. Diagrams
9. Sprint 2 Recap
10. Product Backlog
11. Sprint Summaries
12. Burndown Charts
13. Sprint Retrospective
14. Project Demo

Roles & Responsibilities



Naveen Nayak
Team Lead/ Backend Engineer



Navya Nayak
ML Engineer/ Scrum Master



Madhu Kiran
Frontend Engineer

Roles & Responsibilities



Saranya Chandu
ML Engineer



Aakruthi Reddy
Frontend Engineer

Roles & Responsibilities



Brunai
Developer



Vishwas Mamidi
Developer/ QA

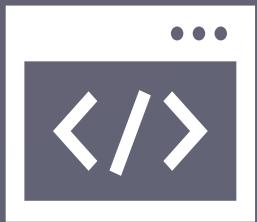
Improvements made from Feedback

- Explained User Story in Detail, Burndown Chart : A detailed walkthrough of each user story, its acceptance criteria, and its progress in the burndown chart.
- All sprint deliverables are submitted to the correct links and wiki page is updated.

Problem Statement

Meetings are essential, but reviewing recordings and notes is time-consuming and often inefficient. Quick Recap solves this problem by providing personalized, AI-powered summaries of meetings, saving users time and maximizing the value of meeting content.

Project Description



WEB APPLICATION
FOR ENHANCED
MEETING
PRODUCTIVITY.



PROVIDES
PERSONALIZED
SUMMARIES OF
MEETING
RECORDINGS.



EMPLOYS NATURAL
LANGUAGE
PROCESSING (NLP)
FOR SUMMARY
GENERATION.



HIGHLIGHTS THE
MOST RELEVANT
INFORMATION
BASED ON USER
PROFILES.



ENABLES USERS TO
QUICKLY GRASP KEY
TAKEAWAYS WITHOUT
REVIEWING LENGTHY
RECORDINGS.

Persona – Project Manager



Name: Isabelle
Age: 48
Gender: Female
Occupation: Project Manager

Isabelle is a high-level executive at a fast-growing tech company. She's intelligent, driven, and constantly juggling multiple projects and responsibilities.

Challenges:

- Attends numerous meetings daily, often back-to-back.
- Struggles to recall key decisions, action items, and strategic discussions from past meetings.
- Lacks time to review lengthy meeting recordings or detailed notes.

Goals:

- Quickly identify action items and delegate tasks effectively.
- Make informed decisions based on accurate and concise meeting information.
- Reclaim valuable time spent reviewing meeting content.

Persona - Journalist



Name: Lena
Age: 28
Gender: Female
Occupation: Journalist

Lena is an investigative journalist who conducts numerous interviews for her stories. She needs to accurately transcribe and analyze these interviews to extract key information and build her narratives.

Challenges:

- Spends countless hours transcribing interviews, which takes away from her research and writing time.
- Requires accurate transcriptions to ensure the integrity of her reporting.
- Finds it difficult to organize and manage a large number of interview recordings.

Goals:

- Quickly and accurately transcribe interviews.
- Improve her research and writing workflow..
- Efficiently extract key quotes and information.

Persona - Student



Name: Carlos
Age: 22
Gender: Male
Occupation: Engineering Student

Carlos is a highly motivated university student pursuing a demanding degree in engineering. He's dedicated to his studies but often finds himself overwhelmed by the volume of information presented in lectures and seminars.

Challenges:

- Struggles to keep up with the fast pace of lectures and take comprehensive notes.
- Finds it difficult to identify the most important concepts and takeaways from lectures.
- Needs a more efficient way to organize and review lecture material.

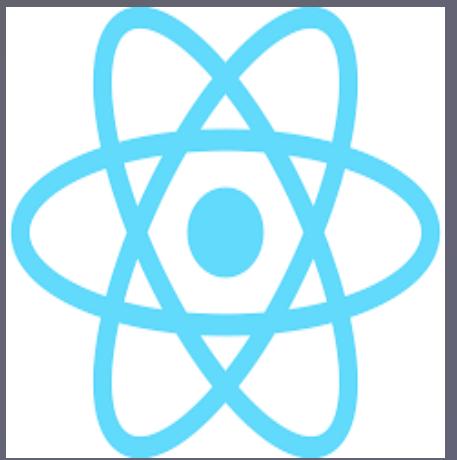
Goals:

- Easily find specific information discussed in lectures.
- Create concise and organized notes from lectures without having to rewatch everything.
- Improve his overall learning efficiency and academic performance.

MVP Product

- Upload audio/video files
- Automatic transcription via speech-to-text
- Basic user profiles (role, interests)
- Personalized summaries based on user profiles
- Core summarization algorithm
- User-friendly web interface.

Technologies



REACT



JavaScript



Python



Firebase



Jira



GitHub

Workflow



Upload



Speech to
text



Personalization



Summarization



Display

Project Schedule

Sprint 0
Feb 11

Sprint 1
Mar 11

Sprint 2
Apr 8

Sprint 3
May 05

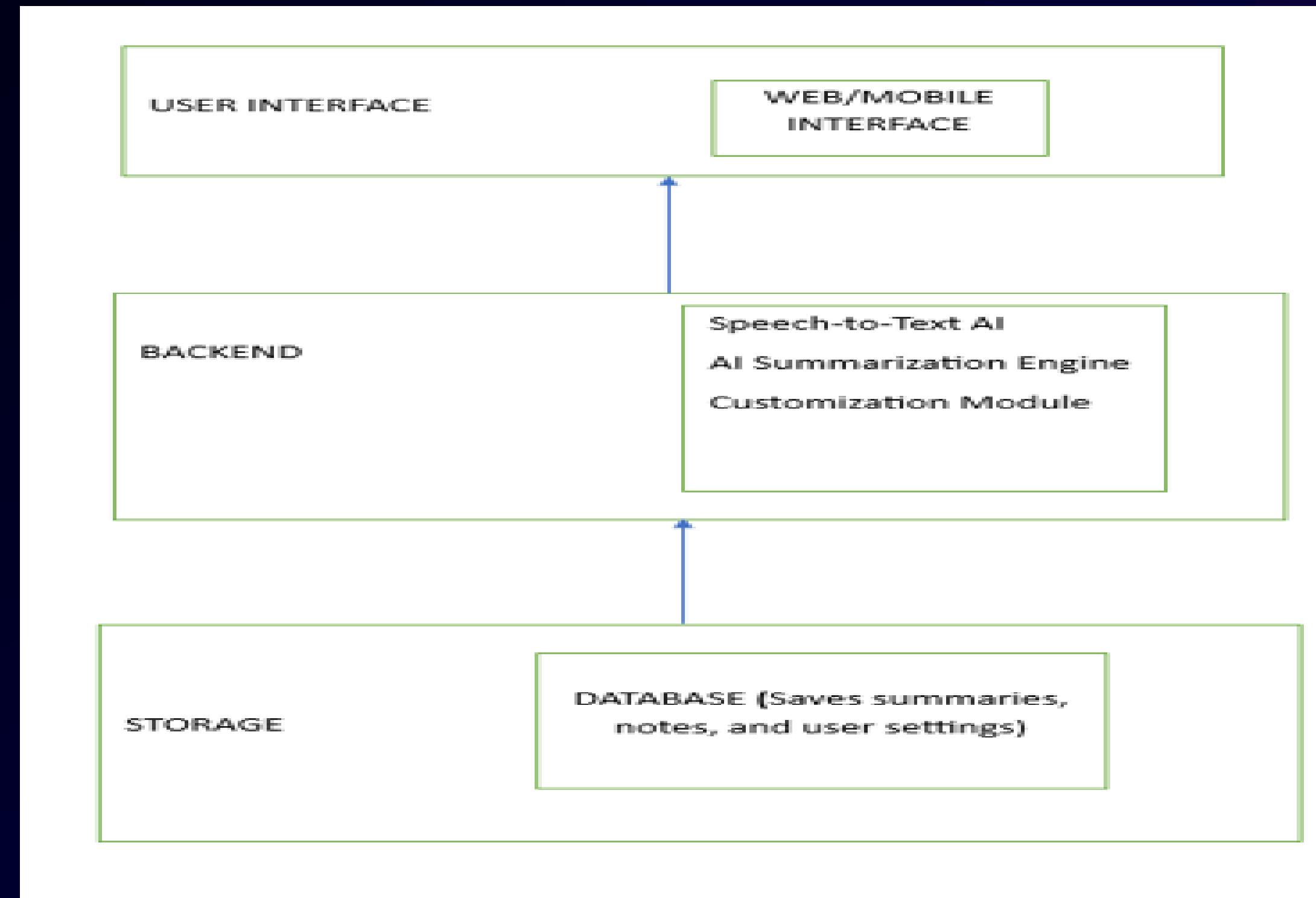
Sprint 0	Sprint 1	Sprint 2	Sprint 3
Daily Scrum call	Daily Scrum call	Daily Scrum call	Daily Scrum call
Sprint Retrospective	Sprint Planning	Sprint Planning	Sprint Planning
	Sprint Retrospective	Sprint Retrospective	Sprint Retrospective
	Product development (Frontend, ML model, Backend)	Product development (Integration of components)	Final Product development

Team Working Agreement

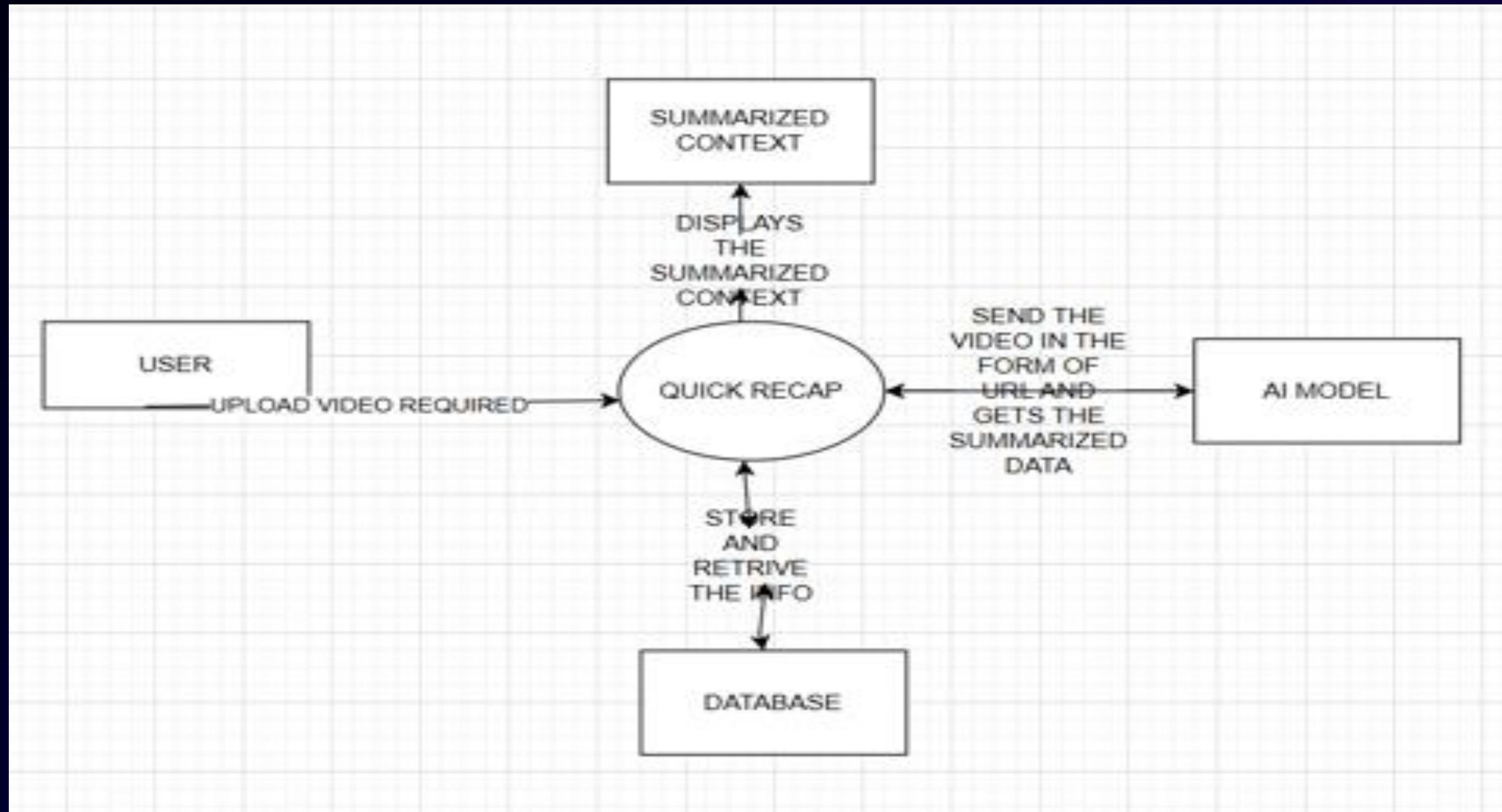
Team Agreement

- We will communicate frequently to track progress, plan for further steps, to solve challenges if required (everyday) but will meet at least thrice a week.
- We will make sure the communication will be respectful.
- Update to other teammates in advance if you are unavailable to attend the meetings (Meeting time will be decided mutually).
- Every person needs to be active, dedicated and need to respond promptly regarding project updates.
- Tasks need to be completed within the time and help to teammates when possible.
- If tasks can't be completed, notify in advance so we can work together to find a solution.
- All decisions will be discussed with the team before being finalized.
- If any issue occurs, based on the majority votes and after discussing with everyone the decision will be taken.
- We encourage everyone to share their ideas and thoughts to build a good project.
- Feedback is encouraged and should be taken positively to improve team performance.
- Members are encouraged to help early instead of waiting until the last minute (if required).
- Disagreements should be resolved within the team in respectful manner.
- Tasks will be tracked by using Jira.
- Weekly Progress updates will be shared within the team to ensure we stay on track.

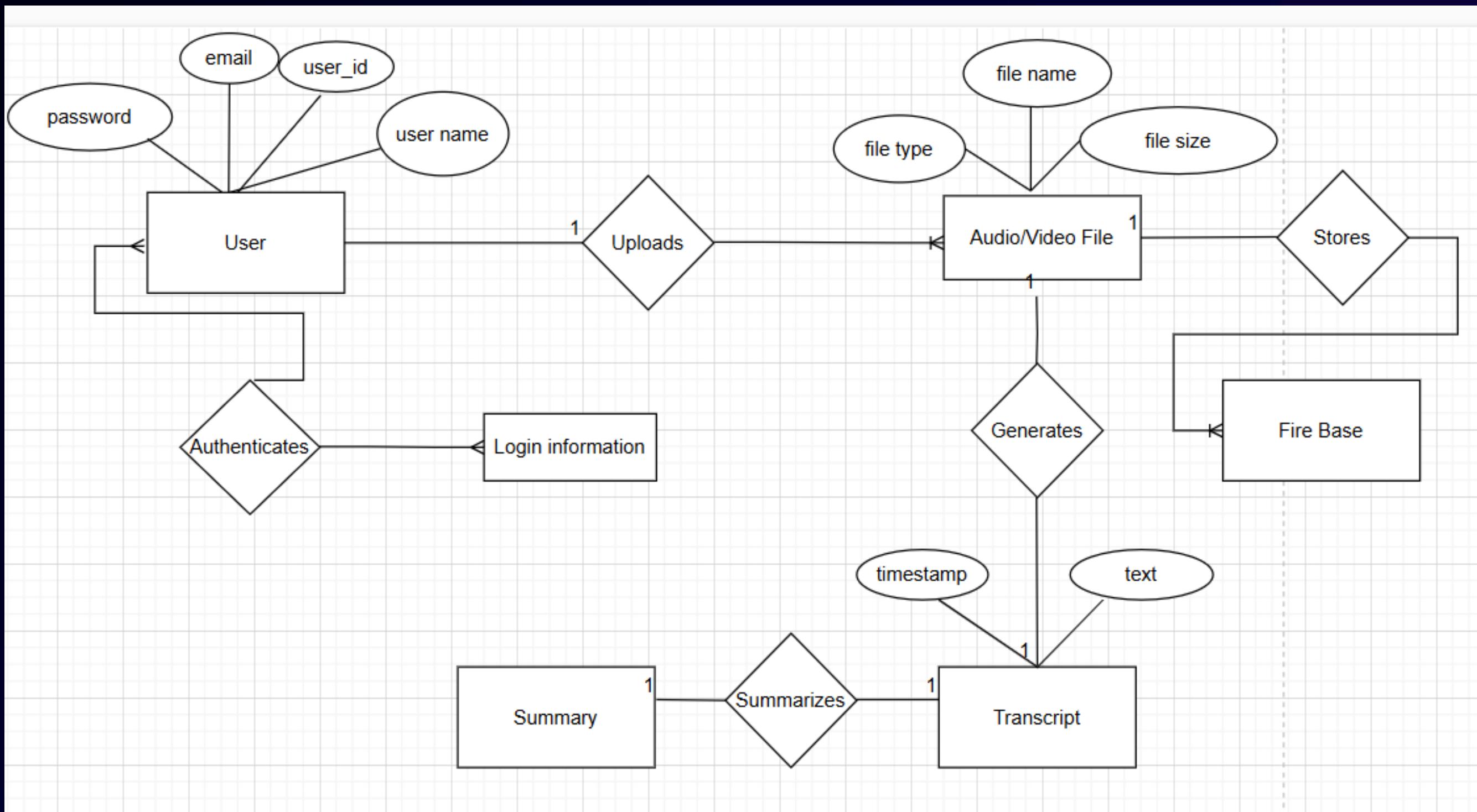
Architecture Diagram



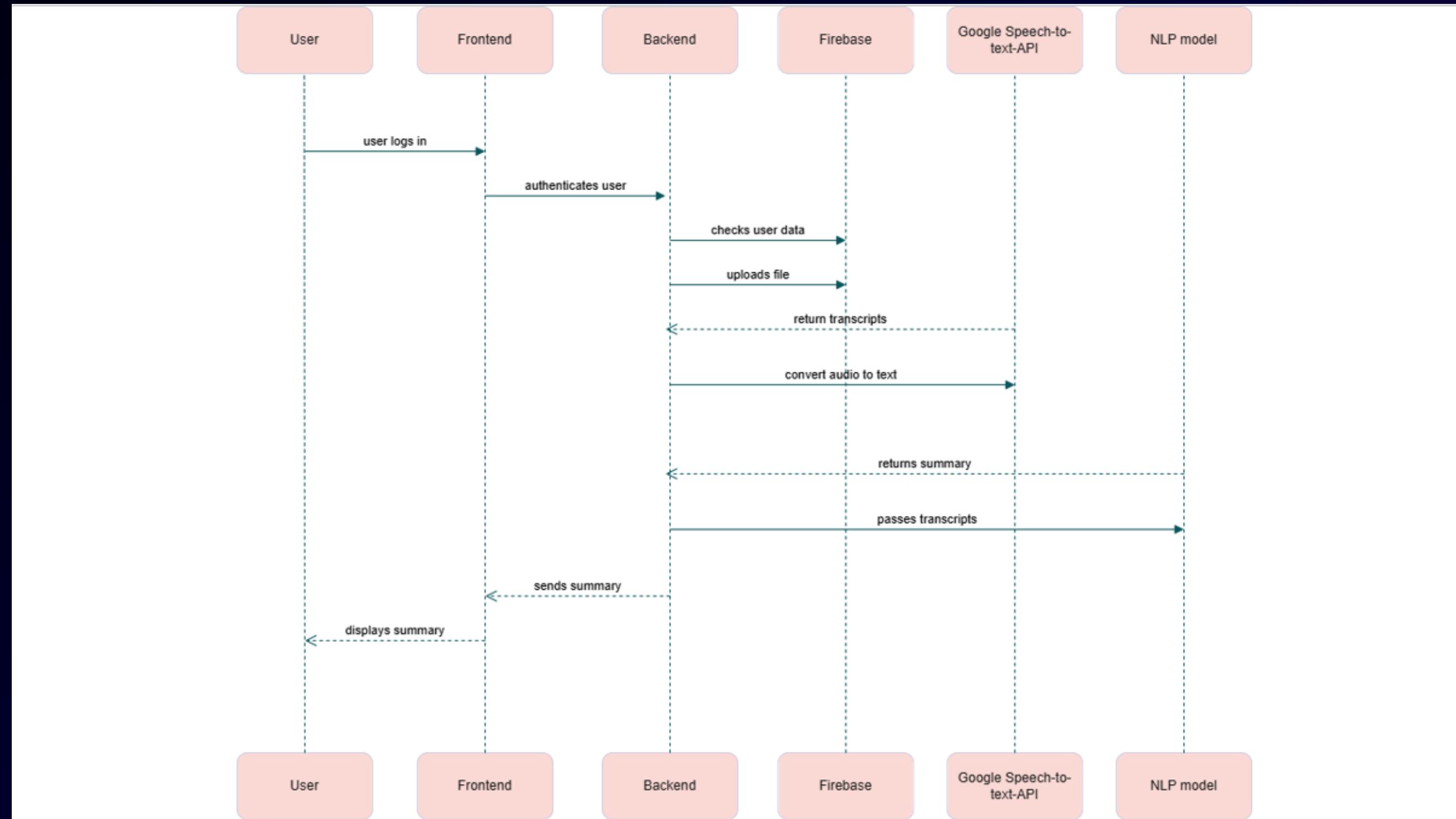
Context Diagram



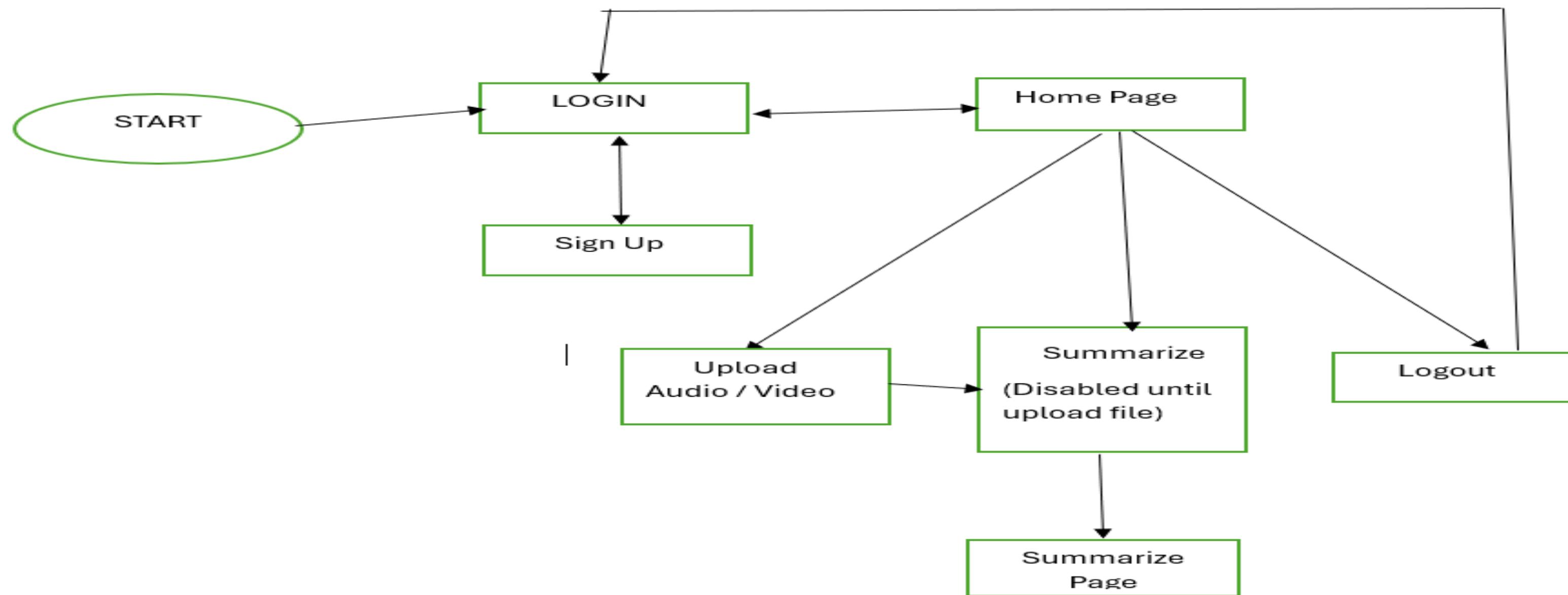
ER Diagram



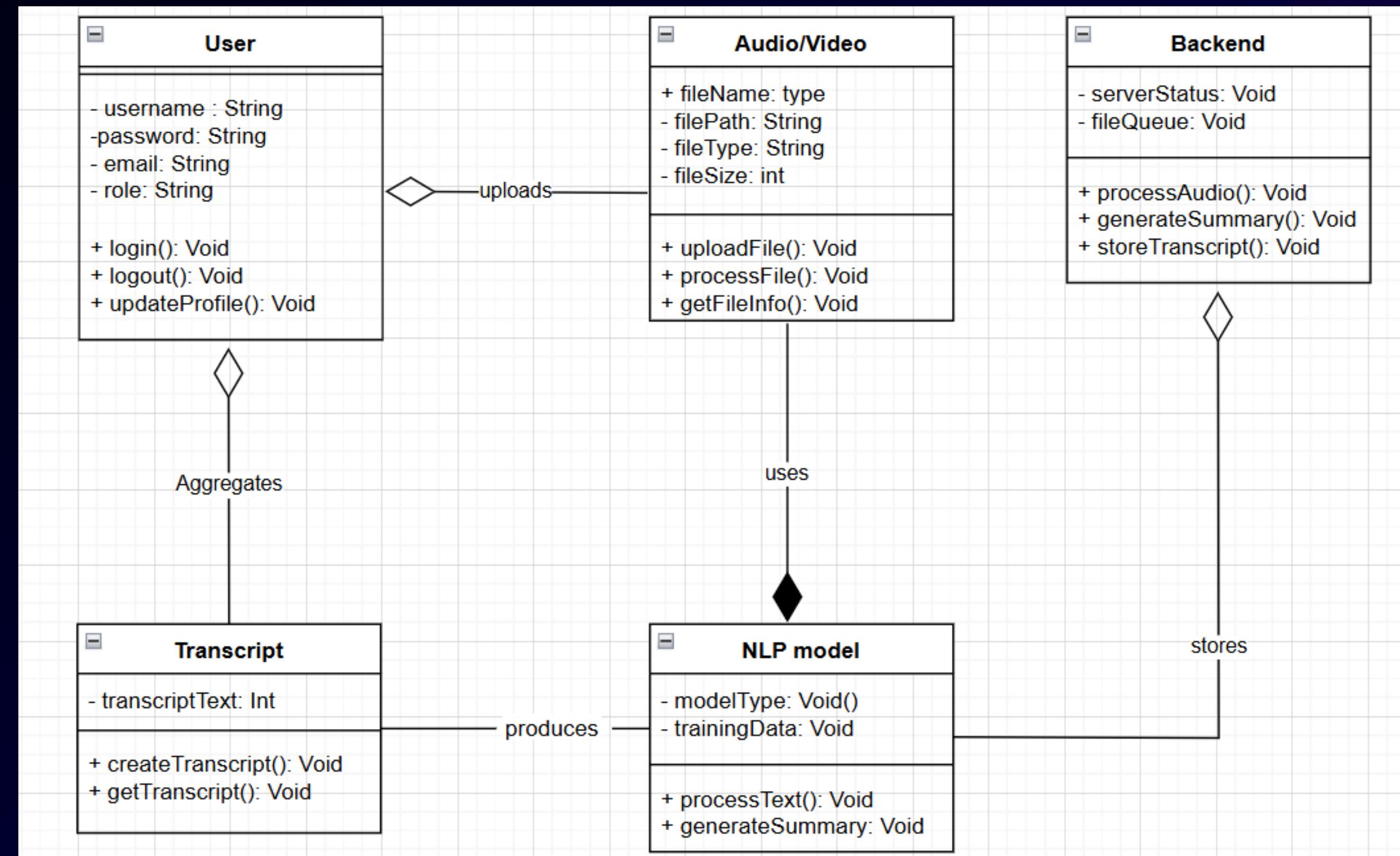
Sequence Diagram



State Diagram



Class Diagram



Sprint 2 Recap

- Audio Transcription : Developed and tested automatic transcription of uploaded audio files into text.
- Video preview feature was implemented in this sprint.
- Implemented summary generation for audio and video transcription.
- Improved User Interface.

Product Backlog

US_ID	User Story	Description	Acceptance Criteria	Feature	Story Points
US_09	Personalized Summary	As a student, I want to get a summary based on my role so that the content is relevant and useful to me.	User can select role from predefined options. Summary output is tailored to the selected role, with detail, and focus adjusted accordingly.	Personalization	8
US_10	Notifications for Summary Completion	As a journalist, I want to receive a notification when my summary is ready, so that I don't have to keep checking manually.	A notification is displayed in the UI when the summary generation process is complete	Notifications	5
US_12	Support for Larger Files	As a journalist, I want to upload larger files without issues, so that I can summarize long-form content without interruptions.	Upload progress is visible to the user with clear feedback (e.g., progress bar). No crashes or timeouts during upload.	File Upload	3

Product Backlog

US_ID	User Story	Description	Acceptance Criteria	Feature	Story Points
US_13	Share Summary	As a project manager, I want to share my summary via email or a link so that I can easily share it with others.	The system includes a share button and a copy link function, allowing the user to easily share the summary via email.	Sharing	5
US_14	Export Summary	As a student, I want to download my summary as a PDF file so that I can save or print it for future reference.	The system should offer a download button that allows the user to download the summary in PDF format with accurate formatting and functionality.	Exporting	5

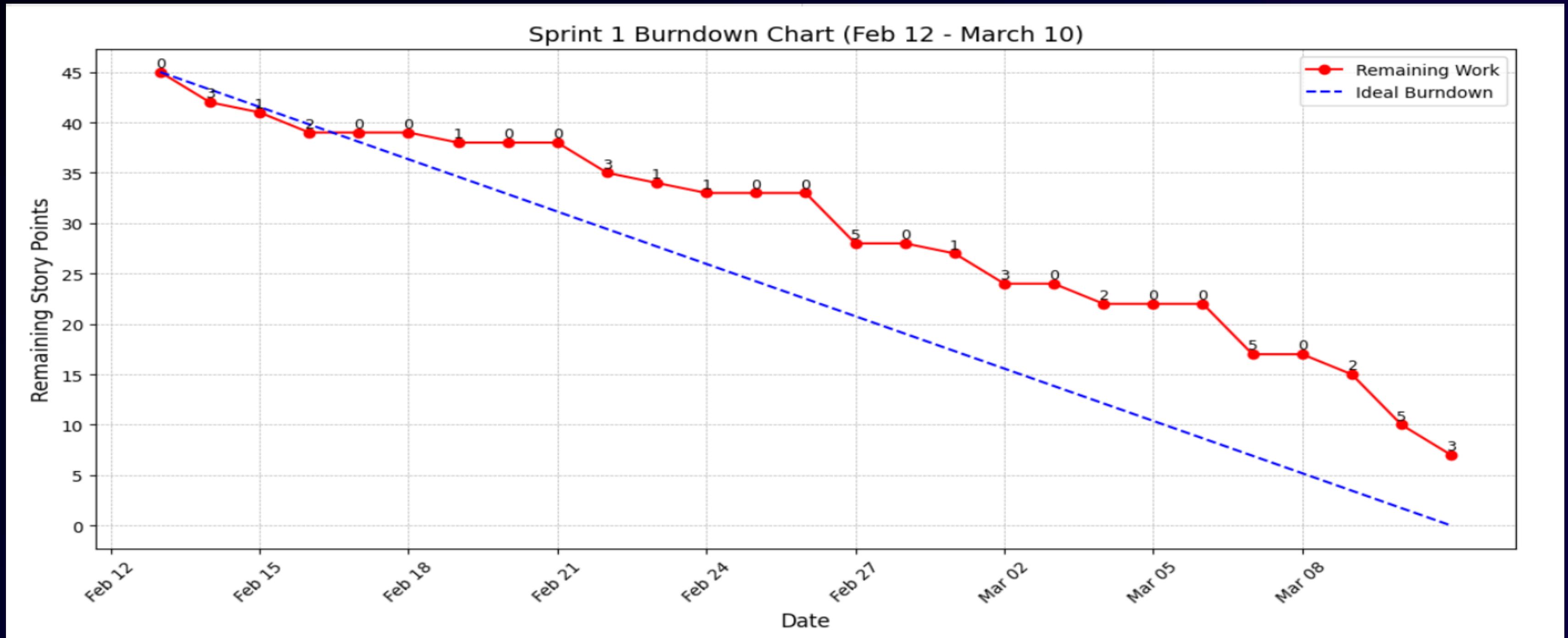
Sprint 1 Summary

Sprint 1 User Stories

US_ID	User Story	Description	Acceptance Criteria	Feature	Story Points
US_01	Upload File	As a student, I want to upload an audio/video file so that I can generate a summary based on its content.	The system allows users to select a file for upload. The UI displays real-time upload status to indicate progress and completion.	File Upload	5
US_02	Transcribe Audio	As a student, I want the system to automatically transcribe the audio I upload into text, so I can easily access and review	The system allows students to upload audio files, automatically transcribes it into text and displays the transcription	Transcription	18
US_03	Handle User Authentication	As a Manager, I want to be able to register, log in, and log out of my account so that I can securely access and manage my content.	The system allows users to register using a username and password. Invalid credentials trigger error messages.	Authentication	5
US_04	File Upload Progress	As a content creator, I want to see the progress of my file upload so that I know the current status and can confirm when it's complete.	The system displays a progress bar indicating real-time upload status. If the upload fails, an error message is shown to the user.	File Upload	3

Sprint 1 Summary

Sprint 1 Burndown Chart



Sprint 2 Summary

Sprint 2 User Stories

US_ID	User Story	Description	Acceptance Criteria	Feature	Story Points
US_05	Improve UI Design	As a student, I want a clean and user-friendly interface, so that I can easily navigate and interact with the application without confusion.	The user interface is responsive, intuitive, and easy to navigate, ensuring a smooth and accessible user experience.	UI/UX	5
US_06	Handle Different Formats	As a journalist, I want the application to support different file types (MP3, MP4, WAV), so that I can upload various audio and video formats for transcription and summarization.	The application supports MP3, MP4, and WAV formats. Displays an error for unsupported formats.	File Upload	3
US_07	Handle User Profile	As a student, I want to be able to view and update my profile information, so that I can keep my details up to date and manage my account effectively.	Users can view and edit username, email, and password. Profile changes are saved	Profile Management	5

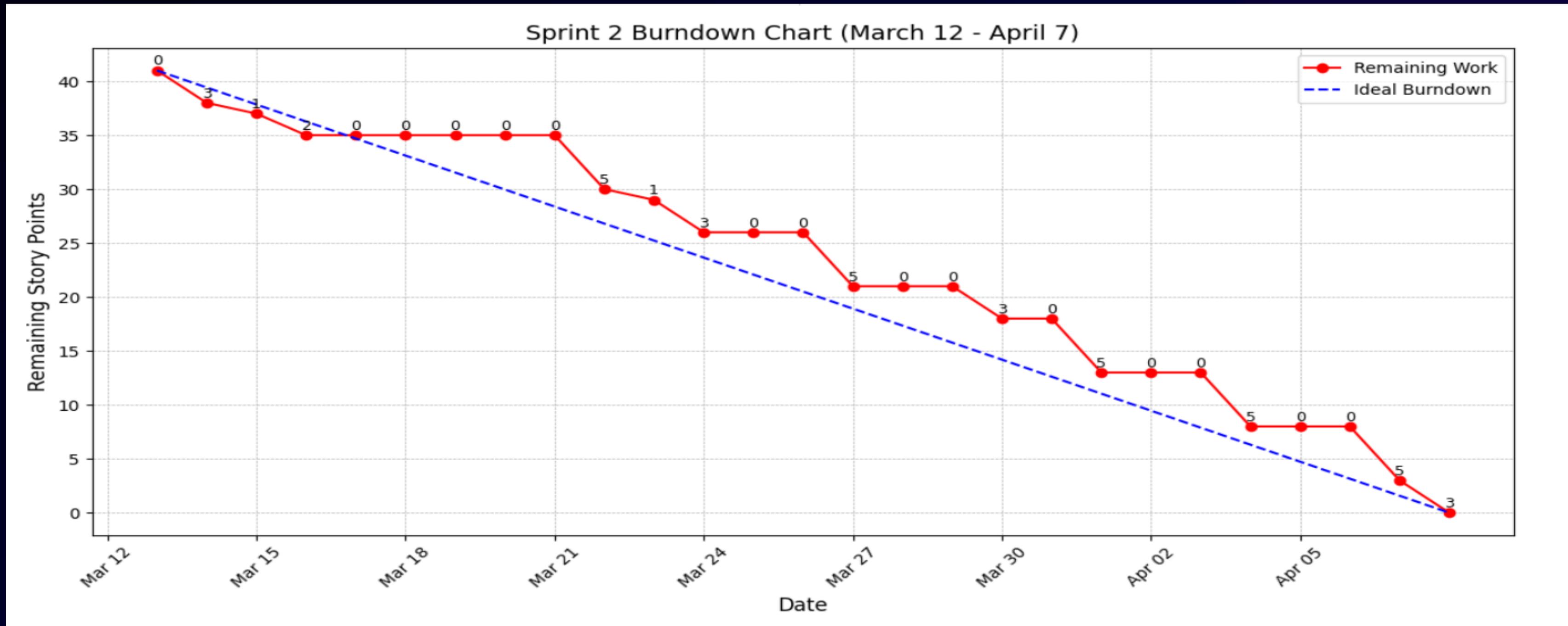
Sprint 2 Summary

Sprint 2 User Stories

US_ID	User Story	Description	Acceptance Criteria	Feature	Story Points
US_08	Generate Summary (Basic)	As a project manager, I want to receive a basic summary of my transcription, so that I can quickly understand the key points of the transcribed content.	Summary highlights the most important points. Summary is concise and easy to read.	Summarization	5
US_11	View Video Previews	As a content creator, I want to view a preview of my uploaded video before generating a summary, so that I can verify the content before proceeding with the summary generation.	Provides a thumbnail or play button for video preview. Users can click on it to view the video directly in the app.	Media Management	5

Sprint 2 Summary

Sprint 2 Burndown Chart



Sprint 3 User Stories

US_ID	User Story	Description	Acceptance Criteria	Feature	Story Points
US_09	Personalized Summary	As a student, I want to get a summary based on my role so that the content is relevant and useful to me.	User can select role from predefined options. Summary output is tailored to the selected role, with detail, and focus adjusted accordingly.	Personalization	8
US_10	Notifications for Summary Completion	As a journalist, I want to receive a notification when my summary is ready, so that I don't have to keep checking manually.	A notification is displayed in the UI when the summary generation process is complete	Notifications	5
US_12	Support for Larger Files	As a journalist, I want to upload larger files without issues, so that I can summarize long-form content without interruptions.	Upload progress is visible to the user with clear feedback (e.g., progress bar). No crashes or timeouts during upload.	File Upload	3

Sprint 3 User Stories

US_ID	User Story	Description	Acceptance Criteria	Feature	Story Points
US_13	Share Summary	As a project manager, I want to share my summary via email or a link so that I can easily share it with others.	The system includes a share button and a copy link function, allowing the user to easily share the summary via email.	Sharing	5
US_14	Export Summary	As a student, I want to download my summary as a PDF file so that I can save or print it for future reference.	The system should offer a download button that allows the user to download the summary in PDF format with accurate formatting and functionality.	Exporting	5

Test Cases

User Story 9: Personalized Summary

Test Case ID	Title	Step Description	Expected Result	Actual Result	Execution Status	Execution Date
TC_09_01	Verify Role Selection Dropdown	1. Launch the app.	User should be able to see a list of predefined roles (e.g., Student, content creator, Manager).	User sees a list of predefined roles (e.g., Student, content creator, Manager).	Passed	14-Apr
		2. Navigate to summary section.				
		3. Click the role selection dropdown.				
TC_09_02	Validate Role Selection Functionality	1. Select each role from dropdown (e.g., Student, Instructor).	The selected role should be highlighted, and saved/used for summary generation.	The selected role is highlighted, and saved/used for summary generation.	Passed	14-Apr
TC_09_03	Verify Summary Personalization - Student	1. Select "Student" role. 2. Click "Generate Summary".	Summary should be focused on content relevant to students (e.g., course progress, assignments).	Summary is focused on content relevant to students (e.g., course progress, assignments).	Passed	14-Apr

Test Cases

User Story 9: Personalized Summary

Test Case ID	Title	Step Description	Expected Result	Actual Result	Execution Status	Execution Date
TC_09_04	Verify Summary Personalization - Instructor	1. Select "Instructor" role. 2. Click "Generate Summary".	Summary should be focused on content relevant to instructors (e.g., student performance, grading tasks).	Summary is focused on content relevant to instructors (e.g., student performance, grading tasks).	Passed	14-Apr
TC_09_05	Verify Summary Updates on Role Change	1. Select a role and generate summary. 2. Change the role. 3. Generate summary again.	Summary should change dynamically according to the new selected role.	Summary changes dynamically according to the new selected role.	Passed	14-Apr

Test Cases

User Story 10 : Notifications for Summary Completion

Test Case ID	Title	Step Description	Expected Result	Actual Result	Execution Status	Execution Date
TC_10_01	Verify Notification Appears After Completion	1. Start generating a summary.	A notification should be shown in the UI informing the user that the summary is ready.	A notification is shown in the UI informing the user that the summary is ready.	Passed	16-Apr
		2. Wait for process to complete.				
TC_10_02	Verify Notification Content	1. Generate a summary.	Notification should include clear and concise message (e.g., "Your summary is now ready").	Notification includes clear and concise message (e.g., "Your summary is now ready").	Passed	16-Apr
		2. Observe the notification text when it's shown.				
TC_10_03	Verify Notification Dismissal Behavior	1. Wait for summary notification.	Notification should disappear from the UI upon user interaction (click, tap, or timeout).	Notification disappears from the UI upon user interaction (click, tap, or timeout).	Passed	16-Apr
		2. Click or tap the notification to dismiss it.				

Test Cases

User Story 12: Support for Larger Files

Test Case ID	Title	Step Description	Expected Result	Actual Result	Execution Status	Execution Date
TC_12_01	Verify Upload of Large File	<ol style="list-style-type: none">Select a large file (e.g., 500MB+).Start the upload.	File should upload successfully without crash or timeout.	File uploads successfully without crash or timeout.	Passed	18-Apr
TC_12_02	Check Upload Progress Visibility	<ol style="list-style-type: none">Upload a large file.Observe the UI during upload.	A visible progress bar should indicate (e.g., progress bar or percentage) upload status.	A visible progress indicator (e.g., progress bar or percentage) reflects upload status.	Passed	18-Apr
TC_12_03	Verify UI Responsiveness During Upload	<ol style="list-style-type: none">Start uploading a large file.Interact with other UI elements (e.g., navigate tabs or menus).	UI should remain responsive and functional during the upload process.	UI remains responsive and functional during the upload process.	Passed	18-Apr

Test Cases

User Story 13: Share Summary

Test Case ID	Title	Step Description	Expected Result	Actual Result	Execution Status	Execution Date
TC_13_01	Verify Share Button Functionality	1. Navigate to a generated summary.	Share options (e.g., Email) should be displayed.	Share options (e.g., Email) is displayed.	Passed	20-Apr
		2. Click the "Share" button.				
TC_13_03	Verify Email Sharing Functionality	1. Click "Share via Email".	Email should be sent with a link or attached summary.	Email should be sent with a link or attached summary.	Passed	20-Apr
		2. Enter recipient email and send.				

Test Cases

User Story 14 : Export Summary

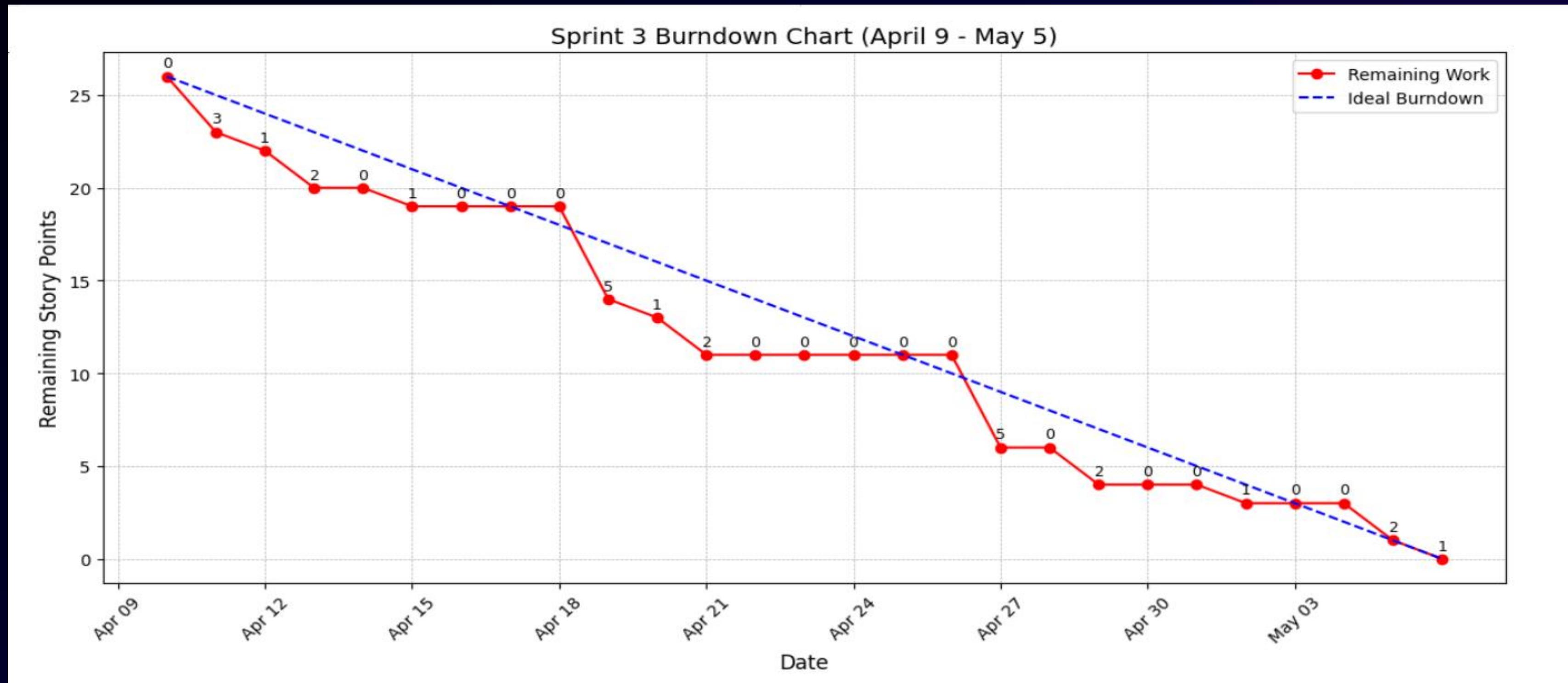
Test Case ID	Title	Step Description	Expected Result	Actual Result	Execution Status	Execution Date
TC_14_01	Verify Download Button Visibility	1. Navigate to a completed summary.	The “Download as PDF” button should be visible and clickable.	The “Download as PDF” button is visible and clickable.	Passed	24-Apr
		2. Look for the “Download as PDF” button.				
TC_14_02	Verify PDF Export Functionality	1. Click the “Download as PDF” button.	PDF should be downloaded successfully and should contain the full summary content.	PDF is downloaded successfully and contains the full summary content.	Passed	24-Apr
		2. Open the downloaded file.				
TC_14_03	Verify PDF Formatting and Layout	1. Open the downloaded PDF.	PDF content should be well-formatted, readable, and matches on-screen layout.	PDF content is well-formatted, readable, and matches on-screen layout.	Passed	24-Apr
		2. Review formatting (e.g., headings, paragraphs, spacing).				

Sprint 3 Stories Completed / Not Completed

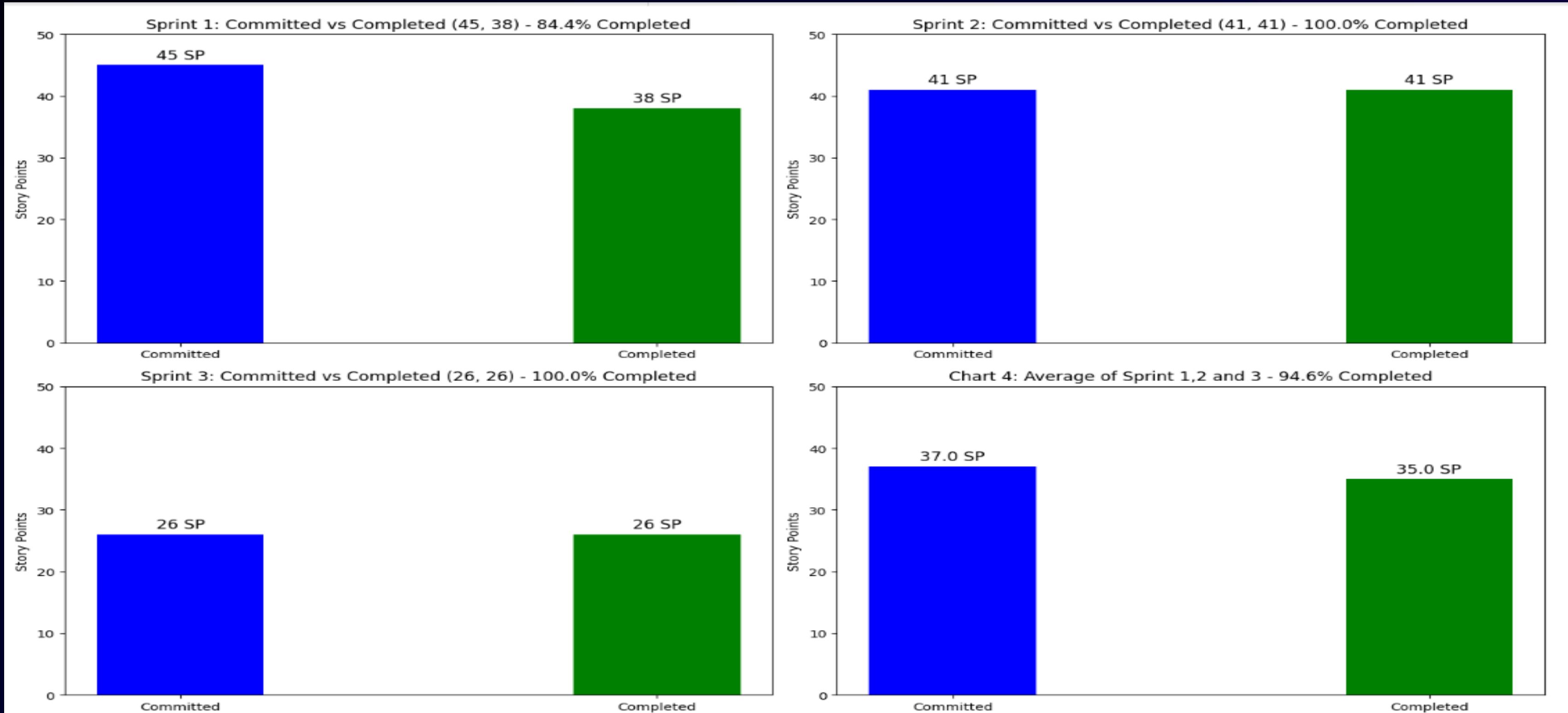
User Stories Completed

US_ID	User Story	Description	Acceptance Criteria	Feature	Story Points
US_09	Personalized Summary	As a student, I want to get a summary based on my role so that the content is relevant and useful to me.	User can select role from predefined options. Summary output is tailored to the selected role, with detail, and focus adjusted accordingly.	Personalization	8
US_10	Notifications for Summary Completion	As a journalist, I want to receive a notification when my summary is ready, so that I don't have to keep checking manually.	A notification is displayed in the UI when the summary generation process is complete	Notifications	5
US_12	Support for Larger Files	As a journalist, I want to upload larger files without issues, so that I can summarize long-form content without interruptions.	Upload progress is visible to the user with clear feedback (e.g., progress bar). No crashes or timeouts during upload.	File Upload	3
US_13	Share Summary	As a project manager, I want to share my summary via email or a link so that I can easily share it with others.	The system includes a share button and a copy link function, allowing the user to easily share the summary via email.	Sharing	5
US_14	Export Summary	As a student, I want to download my summary as a PDF file so that I can save or print it for future reference.	The system should offer a download button that allows the user to download the summary in PDF format with accurate formatting and functionality.	Exporting	5

Metrics - Burndown Chart



Metrics – Team velocity and Completed / Committed Ratio



Sprint 3 - Retrospective

What went well?

- Good communication among team members
- Successfully implemented the key features
- Feedback / action items from previous sprint was worked on

Sprint 3 - Retrospective

What can be improved?

- Jira needs to be updated more frequently
- Communicate regularly and take others help more often. Address issues immediately as they arise
- Ensure that all the sprint deliverables are accessible and updated.

Sprint 3 - Retrospective

Action items

- Assigned to everyone : Update jira as and when there is any progress for the tasks.
- Assigned to everyone : Ensure all sprint deliverables are submitted to the correct links and wiki page is updated.

Sprint 3 - Retrospective

Quick Recap Sprint 3

What went well

Good team communication and collaboration	Action items from previous sprint feedback was worked on in this sprint
+ 0	+ 0
Implemented according to timeline	Team meetings went well

What can be improved

Come with ideas for future work	Verify that all the project wiki page links are working
+ 0	+ 0
Communicate with each other about the pending tasks more often	Update Jira regularly with task progress

Action items

Assigned to all : Verify that all the links on the project wiki page are working and up to date	Assigned to all : Ensure all the sprint deliverables have been submitted to the correct links
+ 0	+ 0

Sprint 3 - Project Demo

QuickRecap

Transform your media into personalized, concise summaries.



Upload Your Media

Easily upload audio or video files. No complicated setup, just upload!



Automatic Transcription

Our technology transcribes your media content into text quickly and accurately.



Personalized Summaries

Get concise, tailored summaries based on your preferences and interests.

[Login](#)[Sign Up](#)

Sprint 3 - Project Demo

QuickRecap

Sign in to continue

Email

Password

Login

New user? [Sign up](#)

Sprint 3 - Project Demo

QuickRecap

Profile Logout

Welcome, Navya Nayak

Upload File: Choose File One o...s.mp4

Upload Video

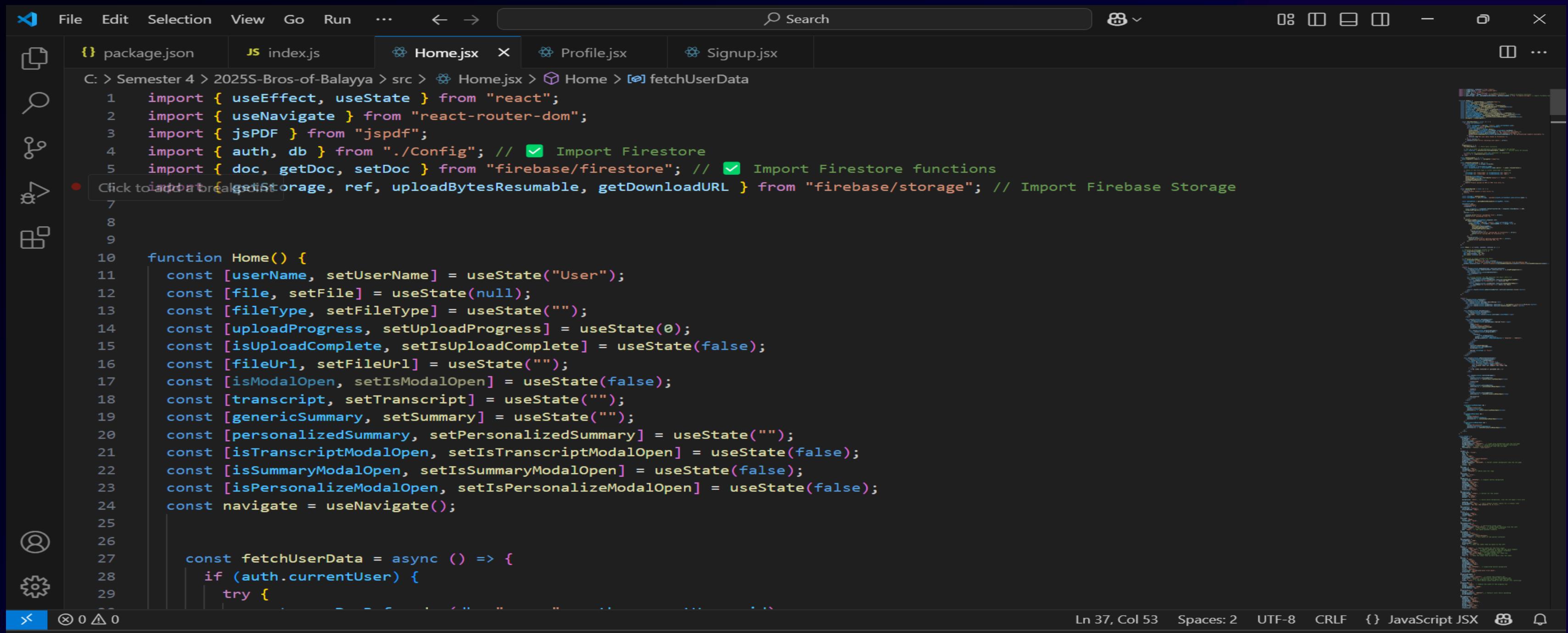


Transcript

Summary

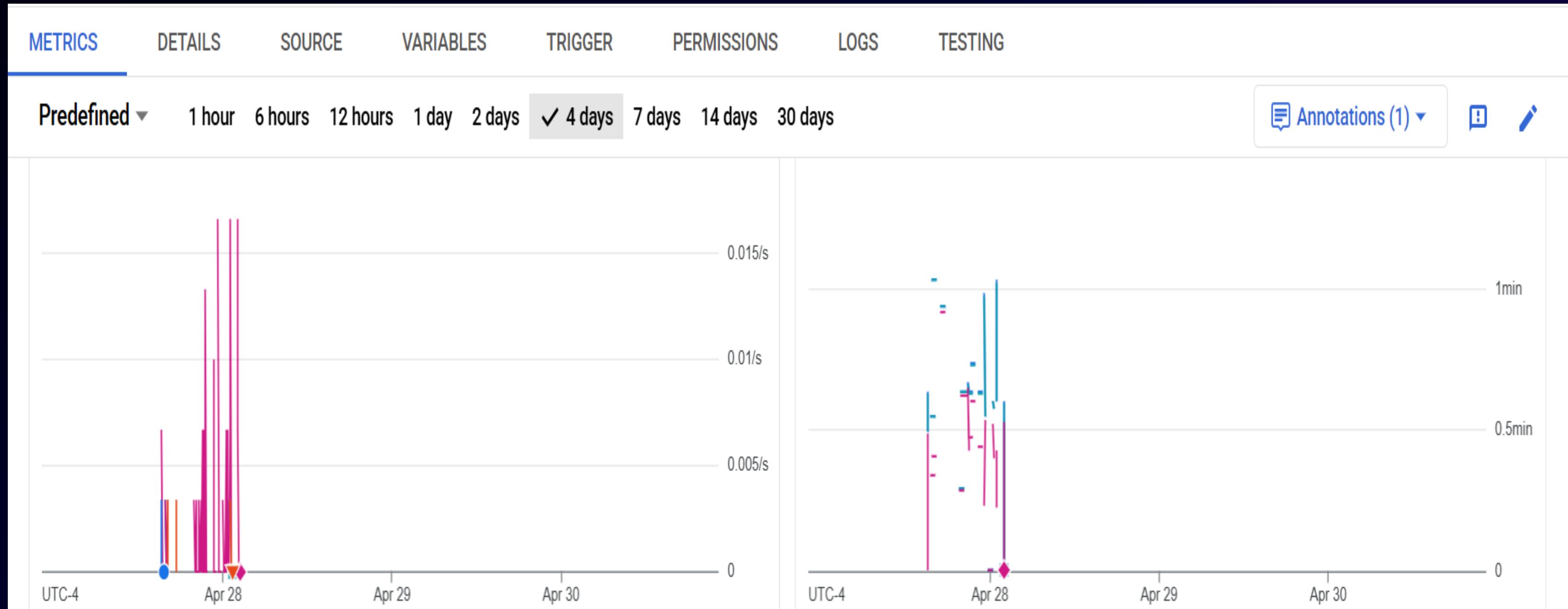
Personalize

Sprint 3 - Project Demo



```
C: > Semester 4 > 2025S-Bros-of-Balayya > src > Home.jsx > Home > fetchUserData
  1 import { useEffect, useState } from "react";
  2 import { useNavigate } from "react-router-dom";
  3 import { jsPDF } from "jspdf";
  4 import { auth, db } from "./Config"; // Import Firestore
  5 import { doc, getDoc, setDoc } from "firebase/firestore"; // Import Firestore functions
  6 import { getStorage, ref, uploadBytesResumable, getDownloadURL } from "firebase/storage"; // Import Firebase Storage
  7
  8
  9
 10 function Home() {
 11   const [userName, setUserName] = useState("User");
 12   const [file, setFile] = useState(null);
 13   const [fileType, setFileType] = useState("");
 14   const [uploadProgress, setUploadProgress] = useState(0);
 15   const [isUploadComplete, setIsUploadComplete] = useState(false);
 16   const [fileUrl, setFileUrl] = useState("");
 17   const [isModalOpen, setIsModalOpen] = useState(false);
 18   const [transcript, setTranscript] = useState("");
 19   const [genericSummary, setSummary] = useState("");
 20   const [personalizedSummary, setPersonalizedSummary] = useState("");
 21   const [isTranscriptModalOpen, setIsTranscriptModalOpen] = useState(false);
 22   const [isSummaryModalOpen, setIsSummaryModalOpen] = useState(false);
 23   const [isPersonalizeModalOpen, setIsPersonalizeModalOpen] = useState(false);
 24   const navigate = useNavigate();
 25
 26
 27   const fetchUserData = async () => {
 28     if (auth.currentUser) {
 29       try {
 30         const userRef = ref(db, `users/${auth.currentUser.uid}`);
 31         const userSnapshot = await getDoc(userRef);
 32         if (userSnapshot.exists()) {
 33           const userData = userSnapshot.data();
 34           setUserName(userData.name);
 35           setFile(userData.profilePic);
 36           setFileType(userData.profilePicType);
 37           setUploadProgress(100);
 38           setIsUploadComplete(true);
 39           setFileUrl(userData.profilePic);
 40         }
 41       } catch (error) {
 42         console.error("Error fetching user data: ", error);
 43       }
 44     }
 45   }
 46
 47   useEffect(() => {
 48     fetchUserData();
 49   }, []);
 50
 51   const handleFileChange = (e) => {
 52     const file = e.target.files[0];
 53     if (file) {
 54       const reader = new FileReader();
 55       reader.onloadend = () => {
 56         setFile(reader.result);
 57         setFileType(file.type);
 58       };
 59       reader.readAsDataURL(file);
 60     }
 61   }
 62
 63   const handleUpload = async () => {
 64     if (!file) return;
 65     const storageRef = ref(getStorage(), `users/${auth.currentUser.uid}/profilePic`);
 66     const uploadTask = uploadBytesResumable(storageRef, file);
 67     uploadTask.on("state_changed", (snapshot) => {
 68       const progress = (snapshot.bytesTransferred / snapshot.totalBytes) * 100;
 69       setUploadProgress(progress);
 70     }, (error) => {
 71       console.error("Error uploading file: ", error);
 72     }, () => {
 73       const downloadURL = getDownloadURL(uploadTask.snapshot.ref);
 74       setFileUrl(downloadURL);
 75       setIsUploadComplete(true);
 76     });
 77   }
 78
 79   const handleLogout = () => {
 80     auth.signOut();
 81   }
 82
 83   const handleModalOpen = (type) => {
 84     setIsModalOpen(true);
 85     if (type === "transcript") {
 86       setTranscript("Transcript content");
 87     } else if (type === "summary") {
 88       setSummary("Summary content");
 89     } else if (type === "personalized") {
 90       setPersonalizedSummary("Personalized content");
 91     }
 92   }
 93
 94   const handleModalClose = () => {
 95     setIsModalOpen(false);
 96   }
 97
 98   const handleTranscriptModalOpen = () => {
 99     setIsTranscriptModalOpen(true);
100   }
101
102   const handleTranscriptModalClose = () => {
103     setIsTranscriptModalOpen(false);
104   }
105
106   const handleSummaryModalOpen = () => {
107     setIsSummaryModalOpen(true);
108   }
109
110   const handleSummaryModalClose = () => {
111     setIsSummaryModalOpen(false);
112   }
113
114   const handlePersonalizeModalOpen = () => {
115     setIsPersonalizeModalOpen(true);
116   }
117
118   const handlePersonalizeModalClose = () => {
119     setIsPersonalizeModalOpen(false);
120   }
121
122   const handleProfilePicChange = (e) => {
123     const file = e.target.files[0];
124     if (file) {
125       const reader = new FileReader();
126       reader.onloadend = () => {
127         setFile(reader.result);
128         setFileType(file.type);
129       };
130       reader.readAsDataURL(file);
131     }
132   }
133
134   const handleProfilePicTypeChange = (e) => {
135     const fileType = e.target.value;
136     setFileType(fileType);
137   }
138
139   const handleProfilePicUpload = async () => {
140     if (!file) return;
141     const storageRef = ref(getStorage(), `users/${auth.currentUser.uid}/profilePic`);
142     const uploadTask = uploadBytesResumable(storageRef, file);
143     uploadTask.on("state_changed", (snapshot) => {
144       const progress = (snapshot.bytesTransferred / snapshot.totalBytes) * 100;
145       setUploadProgress(progress);
146     }, (error) => {
147       console.error("Error uploading profile pic: ", error);
148     }, () => {
149       const downloadURL = getDownloadURL(uploadTask.snapshot.ref);
150       setFileUrl(downloadURL);
151       setIsUploadComplete(true);
152     });
153   }
154
155   const handleProfilePicDelete = () => {
156     if (!file) return;
157     const storageRef = ref(getStorage(), `users/${auth.currentUser.uid}/profilePic`);
158     const deleteTask = deleteObject(storageRef);
159     deleteTask.on("state_changed", (snapshot) => {
160       const progress = (snapshot.bytesTransferred / snapshot.totalBytes) * 100;
161       setUploadProgress(progress);
162     }, (error) => {
163       console.error("Error deleting profile pic: ", error);
164     }, () => {
165       setFile(null);
166       setFileType("");
167     });
168   }
169
170   const handleProfilePicReset = () => {
171     setFile(null);
172     setFileType("");
173   }
174
175   const handleProfilePicTypeReset = () => {
176     setFileType("");
177   }
178
179   const handleProfilePicUploadSuccess = () => {
180     setIsUploadComplete(true);
181   }
182
183   const handleProfilePicDeleteSuccess = () => {
184     setFile(null);
185     setFileType("");
186   }
187
188   const handleProfilePicResetSuccess = () => {
189     setFile(null);
190     setFileType("");
191   }
192
193   const handleProfilePicTypeResetSuccess = () => {
194     setFileType("");
195   }
196
197   const handleProfilePicUploadError = (error) => {
198     console.error("Error uploading profile pic: ", error);
199   }
200
201   const handleProfilePicDeleteError = (error) => {
202     console.error("Error deleting profile pic: ", error);
203   }
204
205   const handleProfilePicResetError = () => {
206     console.error("Error resetting profile pic: ");
207   }
208
209   const handleProfilePicTypeResetError = () => {
210     console.error("Error resetting profile pic type: ");
211   }
212
213   const handleProfilePicUploadProgress = (progress) => {
214     setUploadProgress(progress);
215   }
216
217   const handleProfilePicDeleteProgress = (progress) => {
218     setUploadProgress(progress);
219   }
220
221   const handleProfilePicResetProgress = () => {
222     setUploadProgress(0);
223   }
224
225   const handleProfilePicTypeResetProgress = () => {
226     setUploadProgress(0);
227   }
228
229   const handleProfilePicUploadError = (error) => {
230     console.error("Error uploading profile pic: ", error);
231   }
232
233   const handleProfilePicDeleteError = (error) => {
234     console.error("Error deleting profile pic: ", error);
235   }
236
237   const handleProfilePicResetError = () => {
238     console.error("Error resetting profile pic: ");
239   }
240
241   const handleProfilePicTypeResetError = () => {
242     console.error("Error resetting profile pic type: ");
243   }
244
245   const handleProfilePicUploadProgress = (progress) => {
246     setUploadProgress(progress);
247   }
248
249   const handleProfilePicDeleteProgress = (progress) => {
250     setUploadProgress(progress);
251   }
252
253   const handleProfilePicResetProgress = () => {
254     setUploadProgress(0);
255   }
256
257   const handleProfilePicTypeResetProgress = () => {
258     setUploadProgress(0);
259   }
260
261   const handleProfilePicUploadError = (error) => {
262     console.error("Error uploading profile pic: ", error);
263   }
264
265   const handleProfilePicDeleteError = (error) => {
266     console.error("Error deleting profile pic: ", error);
267   }
268
269   const handleProfilePicResetError = () => {
270     console.error("Error resetting profile pic: ");
271   }
272
273   const handleProfilePicTypeResetError = () => {
274     console.error("Error resetting profile pic type: ");
275   }
276
277   const handleProfilePicUploadProgress = (progress) => {
278     setUploadProgress(progress);
279   }
280
281   const handleProfilePicDeleteProgress = (progress) => {
282     setUploadProgress(progress);
283   }
284
285   const handleProfilePicResetProgress = () => {
286     setUploadProgress(0);
287   }
288
289   const handleProfilePicTypeResetProgress = () => {
290     setUploadProgress(0);
291   }
292
293   const handleProfilePicUploadError = (error) => {
294     console.error("Error uploading profile pic: ", error);
295   }
296
297   const handleProfilePicDeleteError = (error) => {
298     console.error("Error deleting profile pic: ", error);
299   }
300
301   const handleProfilePicResetError = () => {
302     console.error("Error resetting profile pic: ");
303   }
304
305   const handleProfilePicTypeResetError = () => {
306     console.error("Error resetting profile pic type: ");
307   }
308
309   const handleProfilePicUploadProgress = (progress) => {
310     setUploadProgress(progress);
311   }
312
313   const handleProfilePicDeleteProgress = (progress) => {
314     setUploadProgress(progress);
315   }
316
317   const handleProfilePicResetProgress = () => {
318     setUploadProgress(0);
319   }
320
321   const handleProfilePicTypeResetProgress = () => {
322     setUploadProgress(0);
323   }
324
325   const handleProfilePicUploadError = (error) => {
326     console.error("Error uploading profile pic: ", error);
327   }
328
329   const handleProfilePicDeleteError = (error) => {
330     console.error("Error deleting profile pic: ", error);
331   }
332
333   const handleProfilePicResetError = () => {
334     console.error("Error resetting profile pic: ");
335   }
336
337   const handleProfilePicTypeResetError = () => {
338     console.error("Error resetting profile pic type: ");
339   }
340
341   const handleProfilePicUploadProgress = (progress) => {
342     setUploadProgress(progress);
343   }
344
345   const handleProfilePicDeleteProgress = (progress) => {
346     setUploadProgress(progress);
347   }
348
349   const handleProfilePicResetProgress = () => {
350     setUploadProgress(0);
351   }
352
353   const handleProfilePicTypeResetProgress = () => {
354     setUploadProgress(0);
355   }
356
357   const handleProfilePicUploadError = (error) => {
358     console.error("Error uploading profile pic: ", error);
359   }
360
361   const handleProfilePicDeleteError = (error) => {
362     console.error("Error deleting profile pic: ", error);
363   }
364
365   const handleProfilePicResetError = () => {
366     console.error("Error resetting profile pic: ");
367   }
368
369   const handleProfilePicTypeResetError = () => {
370     console.error("Error resetting profile pic type: ");
371   }
372
373   const handleProfilePicUploadProgress = (progress) => {
374     setUploadProgress(progress);
375   }
376
377   const handleProfilePicDeleteProgress = (progress) => {
378     setUploadProgress(progress);
379   }
380
381   const handleProfilePicResetProgress = () => {
382     setUploadProgress(0);
383   }
384
385   const handleProfilePicTypeResetProgress = () => {
386     setUploadProgress(0);
387   }
388
389   const handleProfilePicUploadError = (error) => {
390     console.error("Error uploading profile pic: ", error);
391   }
392
393   const handleProfilePicDeleteError = (error) => {
394     console.error("Error deleting profile pic: ", error);
395   }
396
397   const handleProfilePicResetError = () => {
398     console.error("Error resetting profile pic: ");
399   }
400
401   const handleProfilePicTypeResetError = () => {
402     console.error("Error resetting profile pic type: ");
403   }
404
405   const handleProfilePicUploadProgress = (progress) => {
406     setUploadProgress(progress);
407   }
408
409   const handleProfilePicDeleteProgress = (progress) => {
410     setUploadProgress(progress);
411   }
412
413   const handleProfilePicResetProgress = () => {
414     setUploadProgress(0);
415   }
416
417   const handleProfilePicTypeResetProgress = () => {
418     setUploadProgress(0);
419   }
420
421   const handleProfilePicUploadError = (error) => {
422     console.error("Error uploading profile pic: ", error);
423   }
424
425   const handleProfilePicDeleteError = (error) => {
426     console.error("Error deleting profile pic: ", error);
427   }
428
429   const handleProfilePicResetError = () => {
430     console.error("Error resetting profile pic: ");
431   }
432
433   const handleProfilePicTypeResetError = () => {
434     console.error("Error resetting profile pic type: ");
435   }
436
437   const handleProfilePicUploadProgress = (progress) => {
438     setUploadProgress(progress);
439   }
440
441   const handleProfilePicDeleteProgress = (progress) => {
442     setUploadProgress(progress);
443   }
444
445   const handleProfilePicResetProgress = () => {
446     setUploadProgress(0);
447   }
448
449   const handleProfilePicTypeResetProgress = () => {
450     setUploadProgress(0);
451   }
452
453   const handleProfilePicUploadError = (error) => {
454     console.error("Error uploading profile pic: ", error);
455   }
456
457   const handleProfilePicDeleteError = (error) => {
458     console.error("Error deleting profile pic: ", error);
459   }
460
461   const handleProfilePicResetError = () => {
462     console.error("Error resetting profile pic: ");
463   }
464
465   const handleProfilePicTypeResetError = () => {
466     console.error("Error resetting profile pic type: ");
467   }
468
469   const handleProfilePicUploadProgress = (progress) => {
470     setUploadProgress(progress);
471   }
472
473   const handleProfilePicDeleteProgress = (progress) => {
474     setUploadProgress(progress);
475   }
476
477   const handleProfilePicResetProgress = () => {
478     setUploadProgress(0);
479   }
480
481   const handleProfilePicTypeResetProgress = () => {
482     setUploadProgress(0);
483   }
484
485   const handleProfilePicUploadError = (error) => {
486     console.error("Error uploading profile pic: ", error);
487   }
488
489   const handleProfilePicDeleteError = (error) => {
490     console.error("Error deleting profile pic: ", error);
491   }
492
493   const handleProfilePicResetError = () => {
494     console.error("Error resetting profile pic: ");
495   }
496
497   const handleProfilePicTypeResetError = () => {
498     console.error("Error resetting profile pic type: ");
499   }
500
501   const handleProfilePicUploadProgress = (progress) => {
502     setUploadProgress(progress);
503   }
504
505   const handleProfilePicDeleteProgress = (progress) => {
506     setUploadProgress(progress);
507   }
508
509   const handleProfilePicResetProgress = () => {
510     setUploadProgress(0);
511   }
512
513   const handleProfilePicTypeResetProgress = () => {
514     setUploadProgress(0);
515   }
516
517   const handleProfilePicUploadError = (error) => {
518     console.error("Error uploading profile pic: ", error);
519   }
520
521   const handleProfilePicDeleteError = (error) => {
522     console.error("Error deleting profile pic: ", error);
523   }
524
525   const handleProfilePicResetError = () => {
526     console.error("Error resetting profile pic: ");
527   }
528
529   const handleProfilePicTypeResetError = () => {
530     console.error("Error resetting profile pic type: ");
531   }
532
533   const handleProfilePicUploadProgress = (progress) => {
534     setUploadProgress(progress);
535   }
536
537   const handleProfilePicDeleteProgress = (progress) => {
538     setUploadProgress(progress);
539   }
540
541   const handleProfilePicResetProgress = () => {
542     setUploadProgress(0);
543   }
544
545   const handleProfilePicTypeResetProgress = () => {
546     setUploadProgress(0);
547   }
548
549   const handleProfilePicUploadError = (error) => {
550     console.error("Error uploading profile pic: ", error);
551   }
552
553   const handleProfilePicDeleteError = (error) => {
554     console.error("Error deleting profile pic: ", error);
555   }
556
557   const handleProfilePicResetError = () => {
558     console.error("Error resetting profile pic: ");
559   }
560
561   const handleProfilePicTypeResetError = () => {
562     console.error("Error resetting profile pic type: ");
563   }
564
565   const handleProfilePicUploadProgress = (progress) => {
566     setUploadProgress(progress);
567   }
568
569   const handleProfilePicDeleteProgress = (progress) => {
570     setUploadProgress(progress);
571   }
572
573   const handleProfilePicResetProgress = () => {
574     setUploadProgress(0);
575   }
576
577   const handleProfilePicTypeResetProgress = () => {
578     setUploadProgress(0);
579   }
580
581   const handleProfilePicUploadError = (error) => {
582     console.error("Error uploading profile pic: ", error);
583   }
584
585   const handleProfilePicDeleteError = (error) => {
586     console.error("Error deleting profile pic: ", error);
587   }
588
589   const handleProfilePicResetError = () => {
590     console.error("Error resetting profile pic: ");
591   }
592
593   const handleProfilePicTypeResetError = () => {
594     console.error("Error resetting profile pic type: ");
595   }
596
597   const handleProfilePicUploadProgress = (progress) => {
598     setUploadProgress(progress);
599   }
599
600   const handleProfilePicDeleteProgress = (progress) => {
601     setUploadProgress(progress);
602   }
603
604   const handleProfilePicResetProgress = () => {
605     setUploadProgress(0);
606   }
607
608   const handleProfilePicTypeResetProgress = () => {
609     setUploadProgress(0);
610   }
611
612   const handleProfilePicUploadError = (error) => {
613     console.error("Error uploading profile pic: ", error);
614   }
615
616   const handleProfilePicDeleteError = (error) => {
617     console.error("Error deleting profile pic: ", error);
618   }
619
620   const handleProfilePicResetError = () => {
621     console.error("Error resetting profile pic: ");
622   }
623
624   const handleProfilePicTypeResetError = () => {
625     console.error("Error resetting profile pic type: ");
626   }
627
628   const handleProfilePicUploadProgress = (progress) => {
629     setUploadProgress(progress);
630   }
631
632   const handleProfilePicDeleteProgress = (progress) => {
633     setUploadProgress(progress);
634   }
635
636   const handleProfilePicResetProgress = () => {
637     setUploadProgress(0);
638   }
639
640   const handleProfilePicTypeResetProgress = () => {
641     setUploadProgress(0);
642   }
643
644   const handleProfilePicUploadError = (error) => {
645     console.error("Error uploading profile pic: ", error);
646   }
647
648   const handleProfilePicDeleteError = (error) => {
649     console.error("Error deleting profile pic: ", error);
650   }
651
652   const handleProfilePicResetError = () => {
653     console.error("Error resetting profile pic: ");
654   }
655
656   const handleProfilePicTypeResetError = () => {
657     console.error("Error resetting profile pic type: ");
658   }
659
660   const handleProfilePicUploadProgress = (progress) => {
661     setUploadProgress(progress);
662   }
663
664   const handleProfilePicDeleteProgress = (progress) => {
665     setUploadProgress(progress);
666   }
667
668   const handleProfilePicResetProgress = () => {
669     setUploadProgress(0);
670   }
671
672   const handleProfilePicTypeResetProgress = () => {
673     setUploadProgress(0);
674   }
675
676   const handleProfilePicUploadError = (error) => {
677     console.error("Error uploading profile pic: ", error);
678   }
679
680   const handleProfilePicDeleteError = (error) => {
681     console.error("Error deleting profile pic: ", error);
682   }
683
684   const handleProfilePicResetError = () => {
685     console.error("Error resetting profile pic: ");
686   }
687
688   const handleProfilePicTypeResetError = () => {
689     console.error("Error resetting profile pic type: ");
690   }
691
692   const handleProfilePicUploadProgress = (progress) => {
693     setUploadProgress(progress);
694   }
695
696   const handleProfilePicDeleteProgress = (progress) => {
697     setUploadProgress(progress);
698   }
699
700   const handleProfilePicResetProgress = () => {
701     setUploadProgress(0);
702   }
703
704   const handleProfilePicTypeResetProgress = () => {
705     setUploadProgress(0);
706   }
707
708   const handleProfilePicUploadError = (error) => {
709     console.error("Error uploading profile pic: ", error);
710   }
711
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
```

Sprint 3 - Project Demo



Wiki page Link

To see our progress visit the wiki page below:

<https://github.com/htmw/2025S-Bros-of-Balayya/wiki>

Live project demo

Project Demo YouTube link :

<https://www.youtube.com/watch?v=8LH5HwnJs1I>

Thank You